FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '85

1.1 Write a program to place numbers on a stack with the options of retrieving from or adding to the stack. A Stack is a Last In--First Out (LIFO) type of storing. The user may enter one of three commands: ADD to the stack; TAKE from the stack; or QUIT. If ADD is entered, then the program accepts a value to place on the stack. If TAKE is entered, then the last value placed on the stack is removed and displayed. If QUIT is entered, then the program ends. Example:

```
 INPUT: Enter command: ADD
        Enter number: 80
        Enter command: ADD
        Enter number: 45
        Enter command: ADD
        Enter number: 98
        Enter command: TAKE
OUTPUT: 98
 INPUT: Enter command: TAKE
OUTPUT: 45
 INPUT: Enter command: QUIT
OUTPUT: (program ends)
```

1.2 A teacher writes a set of N consecutive positive integers beginning with 1 on a blackboard. The teacher then erases one of the numbers. Write a program to find the number that was erased if the average of the remaining integers is AV, where N and AV are given as input. Example:

```
 INPUT: Enter N, AV: 11, 4.7

OUTPUT: NUMBER ERASED WAS 9
```

1.3 Write a program to round the square root of a given number at a designated place value and then to sum the digits in the rounded number. The program will accept as input a whole number, N, and an integer between 3 and -4 inclusive for the exponent of the power of 10, D. Let S equal the square root of N rounded to the nearest 10^D, and then display the number S in the form S=####.#### and display the sum of all the digits of S next to the phrase "SUM=". Examples:

```
 INPUT: Enter N, D: 2, -3
OUTPUT: S=   1.4140
        SUM=10

 INPUT: Enter N, D: 625, 1
OUTPUT: S=  30.0000
        SUM=3
```

1.4 The year is 1985. As the U.S.S. RETUPMOC spacecraft whirls through space, Captain James T. Irk notices that the time dial is malfunctioning and is increasing the year every second. He then notices that the years are steadily counting faster and faster. The next thing that he realizes is that the spacecraft has crashed and the time dial reads 2345. Write a program to simulate the behavior of the time dial, displaying the year in the center of the screen. Beginning with the year 1985, increase the count by one year approximately every second for several seconds, and then begin to steadily and rapidly increase the years with less time between each succeeding display until it is counting faster than the eye can comprehend, ending with the year 2345, which remains on the screen. The displays should not flicker. The program should take less than 60 seconds to run.

1.5 In a Tennis tournament there are N number of participants. In the first round there are N/2 or (N-1)/2 games depending on whether N is even or odd respectively. If N is odd then there must be one bye. A bye allows a player to proceed to the second round without playing in the first round. For successive rounds, the number of players is equal to the number of games and byes in the previous round. Write a program that will prompt the user for the number of participants, N, between 2 and 99 inclusive, and will then calculate the number of games and byes starting at the initial round through the final round. For each round and the grand total, print the number of games and byes in the format shown below. Example:

```
 INPUT: Enter N: 27

OUTPUT: ROUND 1   13 GAMES   1 BYE
        ROUND 2    7 GAMES
        ROUND 3    3 GAMES   1 BYE
        ROUND 4    2 GAMES
        ROUND 5    1 GAMES
        TOTAL     26 GAMES   2 BYES
```

1.6 Write a program to find the smallest, the largest, and the sum of all 3-digit numbers, whose 3 digits are non-zero and distinct, between N and M inclusive, where N and M are input as positive integers with N being less than M. Example:

```
 INPUT: Enter N, M: 90, 125

OUTPUT: SMALLEST = 123
        LARGEST = 125
        SUM = 372
```

1.7 Bob Simon of Bob's Cycle Shop wants to improve his billing procedure and decides that a computer program that would calculate values for an invoice and then prints the invoice would be very helpful. Write a program for Bob! Allow him to input the customer's name, product number of part sold, and labor time in hours. Bob charges $10 per hour for labor. The program must prepare an invoice on the screen with the following information: customer name, part # and description, part cost, labor cost, invoice total including 5% sales tax on part cost, rounded to the nearest penny. All amounts must be displaye in the form ###.## as shown below. Bob has seven parts that are to be used in this program.

Part#	Description	Cost
S193	10 INCH SPROCKET	13.95
S867	30 INCH CHAIN	27.50
F234	BLITZ MAG FRAME	119.00
S445	COMPUTCYCLE COMPUTER	33.95
C492	JET BRAKE SET	29.98
J273	27 INCH WHEEL	32.00
T100	27X1 INCH TIRE TUBE	12.50

Example:

```
 INPUT: Enter name: CRAIG
        Enter part#: F234
        Enter time: 10.5

OUTPUT: CUSTOMER NAME: CRAIG
        PART #: F234
        DESCRIPTION: BLITZ MAG FRAME
        PART COST:  119.00
        LABOR COST: 105.00
        5% TAX:       5.95
        TOTAL:      229.95


 INPUT: Enter name: BARBARA
        Enter part#: S193
        Enter time: 4

OUTPUT: CUSTOMER NAME: BARBARA
        PART #: S193
        DESCRIPTION: 10 INCH SPROCKET
        PART COST:   13.95
        LABOR COST:  40.00
        5% TAX:       0.70
        TOTAL:       54.65
```

1.8 Write a program that will prepare a set of labels (on the screen of course), given the number of lines on the label as input (between 3 and 5 inclusive). Each label will have blanks on the first line, the individual's name (last name first, comma and space then first name) on the second line, and the telephone number on the third line. The following names are to be stored in the program either in DATA statements or in arrays:

```
DATA LISA SPINXS, 987-6543
DATA BOB SIMON, 923-4455
DATA BILL SIMON, 123-4567
DATA HARRY TROUTMAN, 876-2174
DATA HARRY PARKER, 222-3333
DATA *END*, 0
```

DATA or array statements will be terminated by "*END*, 0". The program will then alphabetically sort the entries on the basis of last name, then first name. After printing the first three lines of the label, the program is to skip the appropriate number of lines before printing the next label. Example:

```
 INPUT: Enter # of lines on label: 5
OUTPUT:
       PARKER, HARRY
       222-3333



       SIMON, BILL
       123-4567



       SIMON, BOB
       123-4455



       SPINXS, LISA
       987-6543



       TROUTMAN, HARRY
       876-2174
```

1.9 Write a program that will randomly generate a 5x5 matrix of 25 letters, A through Y, centered on the top part of the screen, with every adjacent letter on a row separated by a space. Allow the user to think of a secret letter. The computer must ask the user yes(Y)-or-no(N) questions to logically determine the secret letter (using similar questions as shown in the examples). The computer will start with a score of 11 points and will deduct 1 point for each question that is asked and answered. Display the score in the upper right corner after each question is asked. If the program does not determine the letter before the computer score reaches 0, then no credit is awarded at this time (try again). Possible examples:

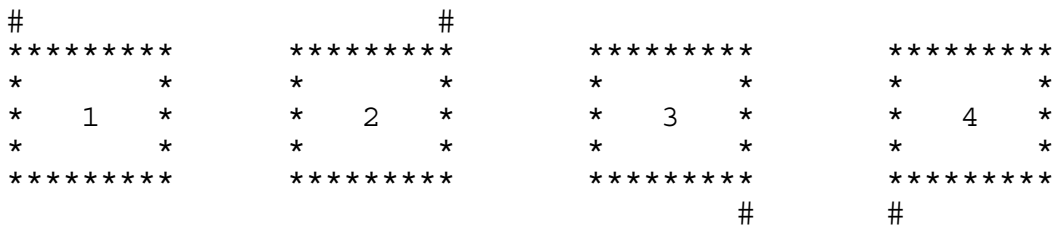
RUN PROGRAM:

```
    OUTPUT:          Q W E R T          SCORE=11
                     Y U I O P
                     A S D F G
                     H J K L M
                     X C V B N

     OUTPUT/INPUT: IS THE LETTER IN ROW 1? N
     OUTPUT: (The score decreases to 10 at the top right)
     OUTPUT/INPUT: IS THE LETTER IN ROW 2? Y
     OUTPUT: (The score decreases to 9)
     OUTPUT/INPUT: IS THE LETTER IN COL 1? N
     OUTPUT: (The score decreases to 8)
     OUTPUT/INPUT: IS THE LETTER IN COL 2? Y
     OUTPUT: (The score decreases to 7)
             YOUR LETTER IS U
```

RUN PROGRAM:

```
    OUTPUT:          X C V B N          SCORE=11
                     M L K J H
                     A S D F G
                     P O I U Y
                     T R E W Q

     OUTPUT/INPUT: IS THE LETTER IN ROW 1? N
     OUTPUT: (The score decreases to 10 at the top right)
     OUTPUT/INPUT: IS THE LETTER IN ROW 2? N
     OUTPUT: (The score decreases to 9)
     OUTPUT/INPUT: IS THE LETTER IN ROW 2? Y
     OUTPUT: (The score decreases to 8)
     OUTPUT/INPUT: IS THE LETTER IN COL 1? N
     OUTPUT: (The score decreases to 7)
     OUTPUT/INPUT: IS THE LETTER IN COL 2? Y
     OUTPUT: (The score decreases to 6)
             YOUR LETTER IS S
```

1.10 Write a program that will allow the user to locate the cursor on the screen by using the keys I,J,K, and L to move up, left, right, and down respectively. Upon pressing 1,2,3, or 4, a square is drawn relative to the cursor's position, if possible, as shown below, using the # symbol to designate the cursor.



If the picture would print off the screen, then display the error message "OFF THE SCREEN" instead. Examples:

RUN PROGRAM: Press the appropriate keys I,J,K,M to place the cursor in the center of the screen.
INPUT: **1**
OUTPUT: The box below with respect to the cursor (#):

```
#
*********
*       *
*   1   *
*       *
*********
```

RUN PROGRAM: Place the cursor in the center of the screen.
INPUT: **3**
OUTPUT:

```
*********
*       *
*   3   *
*       *
*********
        #
```

RUN PROGRAM: Place the cursor at the absolute right side of the screen.
INPUT: **4**
OUTPUT: **OFF THE SCREEN**

RUN PROGRAM: Place the cursor at the bottom of the screen.
INPUT: **2**
OUTPUT: **OFF THE SCREEN**

2.1 Write a program to outline the border of the screen with a random letter; then when the space bar is pressed, the inside border of the new screen will be outlined by a random letter; afterwards, when the space bar is pressed, the inside border of the new screen will be outlined by a random letter, and so on. Continue drawing these rectangles until the whole screen is filled, then press the space bar once again and the screen will clear and start over with a new outer border. A miniature sample run would look like this:

```
RRRRRRRRRR          RRRRRRRRRR          RRRRRRRRRR
R        R          RQQQQQQQQR          RQQQQQQQQR
R        R          RQ      QR          RQYYYYYYYQR
R        R          RQ      QR          RQYYYYYYYQR
R        R          RQQQQQQQQR          RQQQQQQQQR
RRRRRRRRRR          RRRRRRRRRR          RRRRRRRRRR
```

2.2 Write a program to find and print the longest sequence of letters in alphabetical order in a set of N letters entered one at a time. If two or more sequences tie for the longest, then print each of the sequences on successive lines. Each letter must be separated by a space. If a letter repeats, such as the letter "D" in the series of letters A,B,C,D,D,E, then the first sequence is considered to end at the first "D" and the second sequence begins at the second "D". Example:

```
 INPUT: Enter N: 5
        Enter letter: J
        Enter letter: E
        Enter letter: S
        Enter letter: S
        Enter letter: U
OUTPUT: E S
        S U
```

```
 INPUT: Enter N: 7
        Enter letter: S
        Enter letter: I
        Enter letter: S
        Enter letter: L
        Enter letter: O
        Enter letter: R
        Enter letter: D
OUTPUT: L O R
```

2.3 Write a program to simulate a word processor: Allow the program to accept as input a paragraph (one string with no leading spaces and no more than 127 characters in length) and to print the words out on the screen providing a 5-character left margin and no more than 30 characters on a line. Do not split up words. A word is defined as a set of characters in between two spaces (except for the first and last words of the string). The first word of the paragraph is to be indented 5 spaces. The last word of a sentence is followed by a period and 2 spaces (as input), except for the last word of the paragraph which is followed by just a period. New sentences beginning at the left margin are not to be indented by the trailing spaces of the previous sentence. Example:

```
 INPUT: Enter text: DO NOT SPLIT UP WORDS.  THE FIRST WORD OF
                    THE PARAGRAPH IS TO BE INDENTED 5 SPACES.

OUTPUT:          DO NOT SPLIT UP WORDS.
           THE FIRST WORD OF THE
           PARAGRAPH IS TO BE INDENTED 5
           SPACES.
```

2.4 Write a program to accept a word as input and print the word with its consonants alphabetized and placed back into the consonant positions, and likewise have the vowels alphabetized and placed back into the vowel positions. Example:

```
 INPUT: Enter word: ELEPHANT
OUTPUT: AHELNEPT
```

2.5 Write a program that will accept at most 9 words, each having at most 10 characters, and then will check the words for common letters and print those letters that are occur in all the words. The program is to allow the user to choose one of the common letters, then the program is to arrange the words in a list so that the FIRST common letter appears in the same column. If there are no common letters then print the message: NO COMMON LETTERS. Examples:

```
 INPUT: Enter number of words: 3
        Enter word: BROTHER
        Enter word: MOTHER
        Enter word: TUTOR
OUTPUT: O R T
 INPUT: Choose letter: T
OUTPUT: BROTHER
         MOTHER
           TUTOR

 INPUT: Enter number of words: 2
        Enter word: MOM
        Enter word: DAD
OUTPUT: NO COMMON LETTERS
```

2.6 Three local cross country teams compete in a double dual race. Each team consists of seven runners, but only the first five finishers of a team contribute to that team's score. As the runners cross the finish line, gasping for breath, the judge writes the INITIAL of the runner's team name and the NUMBER indicating the runner's finishing position, e.g. 1 for 1st, 2 for 2nd and so on. To find the score for teams A and B and to decide which of the two wins, the scorekeeper temporarily eliminates all of C's positions, and then repositions the runners from A and B into places 1 through 14. The team's score consists of the sum of the places of their first five runners. The lower team score wins. If there is a tie then the team whose sixth runner crossed the finish line first is the winner.

Write a program that computes the score for each pair of three teams and determines the winner of each pair. The program must allow the user to assign all 21 runners' team INITIAL to finishing places. Team initials can be any letter in the alphabet. Example:

INPUT:
Place 1: **A**
Place 2: **B**
Place 3: **A**
Place 4: **B**
Place 5: **A**
Place 6: **B**
Place 7: **C**
Place 8: **C**
Place 9: **C**
Place 10: **C**
Place 11: **B**
Place 12: **A**
Place 13: **C**
Place 14: **B**
Place 15: **C**
Place 16: **B**
Place 17: **A**
Place 18: **A**
Place 19: **C**
Place 20: **B**
Place 21: **A**

Example of repositioning teams A and B:

1: A
2: B
3: A
4: B
5: A
6: B
7: B
8: A
9: B
10: B
11: A
12: A
13: B
14: A

OUTPUT: (in any order)

TEAM A: 28 POINTS
TEAM B: 28 POINTS
TEAM B WINS!

TEAM A: 25 POINTS
TEAM C: 31 POINTS
TEAM A WINS!

TEAM B: 24 POINTS
TEAM C: 31 POINTS
TEAM B WINS!

2.7 Write a program that will make it easy to manipulate tables of numerical data with at most 4 digits and one decimal point. The program must first initialize a 3x3 array with the data shown in the example below. The array will be displayed as a table consisting of three rows with three items in each row. Complete the table so that sums of the rows are found in a fourth column and sums of the columns are found in the fourth row. After the initial table is loaded, the program must display the following menu:

A. EDIT OR CHANGE A VALUE
B. DISPLAY THE RESULTS
C. QUIT

If option (A) is chosen, the user will enter the row and column to be changed, then the new value. The program then returns to the menu upon hitting a key. If option (B) is chosen, the table will be displayed; Both the numbers and the sums are to be displayed in the form ###.## and separated from each other by two spaces. The program then returns to the menu upon hitting a key. If option (C) is chosen, the program will terminate. Example:

RUN PROGRAM:

```
OUTPUT: A. EDIT OR CHANGE A VALUE
        B. DISPLAY THE RESULTS
        C. QUIT
 INPUT: Enter option: B
OUTPUT:  10.11   20.22   30.33   60.66
         11.10   22.20   33.30   66.60
         10.00   20.00   30.00   60.00
         31.21   62.42   93.63  187.26
 INPUT: (press any key)
OUTPUT: A. EDIT OR CHANGE A VALUE
        B. DISPLAY THE RESULTS
        C. QUIT
 INPUT: Enter option: A
        Enter row, col: 1, 2
        Enter number: 60.55
 INPUT: (press any key)
OUTPUT: A. EDIT OR CHANGE A VALUE
        B. DISPLAY THE RESULTS
        C. QUIT
 INPUT: Enter option: B
OUTPUT:  10.11   60.55   30.33  100.99
         11.10   22.20   33.30   66.60
         10.00   20.00   30.00   60.00
         31.21  102.75   93.63  227.59
 INPUT: (press any key)
OUTPUT: A. EDIT OR CHANGE A VALUE
        B. DISPLAY THE RESULTS
        C. QUIT
 INPUT: Enter option: C
OUTPUT: (program terminates)
```

2.8 Write a program that checks all combinations of four different whole numbers between 0 and 9 inclusive to see if the product of any two can be expressed as a 2-digit numeral using the other two numbers as digits. The program must print the combinations for which the condition holds true along with the illustration. Each set must be displayed with its elements in the following order: smaller multiplicand, larger multiplicand, digits of product. Amongst the other sets, each set must be displayed in ascending order when considering the smaller and larger multiplicands together. In the example below, set 2,5,1,0 comes before 2,7,1,4 because 25 comes before 27. Similarly, 27 comes before 28 in the set 2,8,1,6. A final total must also be displayed in the form shown below. Partial example:

RUN PROGRAM:

```
OUTPUT: 2  5  1  0    2 X 5 = 10
        2  7  1  4    2 X 7 = 14
        2  8  1  6    2 X 8 = 16
        :  :  :  :    :       :
        :  :  :  :    :       :
        8  9  7  2    8 X 9 = 72
        TOTAL = ##
```

2.9 Write a program to accept N words, after N is input as a number less than 15, and then to accept a word with a wildcard, "*". The wildcard could come before, after, or in between the letters of the word. Your program must then display all words that were input that match the format of the wildcard expression, where the wildcard could represent 0 to 10 letters. All words are displayed in the order in which they were input. If no words are found then display the message "NO WORDS FOUND". The program is to continue to accept as input a wildcard word until a word is entered without a wildcard. Example:

```
 INPUT: Enter N: 3
        Enter word: RUN
        Enter word: RUNNING
        Enter word: RUNS
        Enter string: RUN*
OUTPUT: RUN
        RUNNING
        RUNS
 INPUT: Enter string: *UN
OUTPUT: RUN
 INPUT: Enter string: R*N
OUTPUT: RUN
 INPUT: Enter string: *INGS
OUTPUT: NO WORDS FOUND
 INPUT: Enter string: END
OUTPUT: (program terminates)
```

2.10 A particular building has three major sections, an office section, a computer section, and a dry storage section. Each section's temperature is controlled by a central processing unit for different maximums and minimums as set by a section's thermostat. Heat is generated in the office section so that the temperature rises at a rate of .1 degree Fahrenheit (F) every 15 seconds. Heat is generated at the rate of .2(F) every 15 seconds in the computer room. The rate in dry storage is .1(F) every minute. The central air conditioning unit has the capacity to cool any one section at the rate of .1(F) every 3 seconds if no heat is being generated. If 2 sections are being cooled the air conditioner will cool at the rate of .1(F) every 6 seconds and for all 3 sections the rate is .1(F) every 9 seconds.

Write a program for the controller that will monitor the temperatures in each section every 15 seconds, alert the controller to turn air conditioning off or on in a section when necessary and will print a status report every 5 minutes or when the air conditioner is turned off or on in any section. The status report must include the sections that the air conditioning is turned off or on (denoted by 0 or 1 respectively), the temperature in each room (rounded to the nearest tenth of a degree), and the time of the status report (of the form Minutes:Seconds). Assume that heat is being generated at the same time that a section is being cooled. Initialize the program with minimum temperatures for each section and the air conditioning off. Air conditioning is changed (turned on or off) when it exceeds or precedes its maximum or minimum temperature respectively during a certain interval of time. For example if the computer room reads 64.8(F) and the air conditioning is currently off, then a message should not be displayed to turn the air conditioning off again for that section if the temperature reads 64.9(F) fifteen seconds later. The program ends when it reaches the last 5-minute interval, entered by the user.

Minimum and Maximum Acceptable Temperatures
Office 72.0F to 78.0F
Computer Rm 65.0F to 70.0F
Dry Storage 79.0F to 85.0F

Example:

INPUT: Enter last 5-minutes: **30**

OUTPUT:

OF	CO	DS	OFFICE	COMP.	DRY.	MIN:SEC
0	0	0	72.0	65.0	79.0	0:00
0	0	0	74.0	69.0	79.5	5:00
0	1	0	74.6	70.2	79.7	6:30
0	1	0	76.0	66.0	80.0	10:00
0	0	0	76.4	64.8	80.1	11:00
0	0	0	78.0	68.0	80.5	15:00
1	0	0	78.1	68.2	80.5	15:15
1	1	0	74.1	70.2	80.8	17:45
1	1	0	72.7	69.7	81.0	20:00
0	1	0	72.0	69.5	81.1	21:15
0	0	0	73.5	65.0	81.5	25:00
0	0	0	75.5	69.0	82.0	30:00

3.1 A standard six sided die has a 1 on the opposite side of the 6, and a 5 on the opposite side of the 2. When the 1 is on the top and the 5 is in front, the 4 is on the right side, and the 3 is on the left side. Write a program that, accepts as input, the numbers on the top and front sides of the die and prints out the numbers on all six sides of the die clearly labeled. Example:

```
 INPUT: Enter top, front: 1, 3
OUTPUT: TOP=1  FRONT=3  RIGHT=5
        BACK=4  LEFT=2  BOTTOM=6
```

3.2 Write a program to factor a quadratic equation with integral coefficients and rational roots. Input will be A,B,C of Ax^2+Bx+C, where A is greater than 0 and C is not equal to 0. Output should be of this form: Q(Rx+S)(Tx+U), where Q,R,S,T, and U are integers and R,Q are positive. For example 4x^2+2x-12 should yield 2(x+2)(2x-3) or 2(2x-3)(x+2), order does not matter. Note that R=1 is omitted and that there is a '-' sign instead of '+-'. If Q=1 then Q must not be printed. If the quadratic equation cannot be factored then display the message: CANNOT BE FACTORED.

```
 INPUT: Enter A, B, C: 30, 4, -2
OUTPUT: 2(5X-1)(3X+1)

 INPUT: Enter A, B, C: 2, 4, 6
OUTPUT: CANNOT BE FACTORED
```

3.3 Write a program to simulate a calculator. Input will be a mathematical expression using whole numbers (less than 1000) and the symbols +,-,*,/. By following the rules of algebraic order, compute the numerical value of the expression and display it in the form ###.###. Examples:

```
 INPUT: Enter expression: 3+4*2-15
OUTPUT:  -4.000

 INPUT: Enter expression: 250*4-50*5+100/3
OUTPUT: 783.333
```

3.4 Write a program to print out all the digits of N factorial (N!), once given N as a whole number less than 50. The output displayed must contain all the digits in the resulting factorial number and not an abbreviated form using exponential notation (i.e. 2.432902E+18).

```
 INPUT: Enter N: 20
OUTPUT: 2432902008176640000
```

3.5 Write a program that will print the sum and difference (clearly labeled) of any two positive decimal numbers of up to 30 digits each with the decimal point between any two of the digits. Since the first number input will be numerically larger than the second number input, the difference output will always be positive. Example:

```
 INPUT: Enter #1: 987654321.123456789
        Enter #2: 123456789.987654321

OUTPUT: SUM = 1111111111.111111110
        DIFFERENCE = 864197531.135802468
```

3.6 Write a program to print a snake (a trail of 30 asterisks '*') centered on the screen. Upon hitting appropriate keys (like I,J,K,M or something similar), the snake's head moves in the appropriate direction while the rest of the snake slithers along the same right angle paths. The snake is to move CONTINUOUSLY in the designated direction UNTIL a new directional key is hit. The snake will be 30 asterisks long throughout the entire run; Do not leave a sketched path. The snake cannot go backwards, e.g. if it is going to the right its next direction cannot be to the left. The snake continues moving until it runs into itself or it runs off the screen or a non-directional key is pressed.

3.7 Write an EFFICIENT program to accept as input a word with at most 7 distinct letters and a positive integer K. Your program is to print out the Kth, 2*Kth, and 3*Kth elements of the alphabetized list of all the permutations of the word. For example if the word CAT is entered with K=2 then the computer would form the list ACT, ATC, CAT, CTA, TAC, TCA, and output would be: ATC, CTA, TCA. Example:

```
 INPUT: Enter word: CAT
        Enter K: 2
OUTPUT: ATC  CTA  TCA
```

3.8 Suppose you had a playing board with N rows with N squares in each row. You are to place pennies on the squares so that each row, column, and main diagonal has at most one penny. Write a program that will accept as input a positive number N, between 3 and 14 inclusive, and will print the largest possible number of pennies that can be placed on the board. The program also prints one of many solutions to the puzzle, displaying the units digit of the N numbers horizontally, each separated by a space, and the units digit for the N numbers vertically; Place an asterisk in the position where a penny can be placed and display the coordinates of each penny to the right with the sum of the coordinates. Examples:

```
 INPUT: Enter N: 3

OUTPUT: TOTAL = 3
          1 2 3
        1 *       (1,1) SUM = 2
        2     *   (2,3) SUM = 5
        3   *     (3,2) SUM = 5

 INPUT: Enter N: 4

OUTPUT: TOTAL = 4
          1 2 3 4
        1   *       (1,2) SUM = 3
        2       *   (2,4) SUM = 6
        3 *         (3,1) SUM = 4
        4     *     (4,3) SUM = 7
```

3.9 Write a program to determine the minimum number of moves required to move a stack of N blocks having graduated sizes to one of two alternate locations. (There are three locations all together.) N will be input as a positive integer less than 16. When moving blocks, you can only move one block at a time and you cannot stack a larger block on a smaller block. When finished, the blocks in the new stack will be in the same order as they were in the beginning. Your program could be written in a very few program lines if you can develop an algorithm for determining the required number of moves. For example, the following 7 moves will transfer the three graduated blocks from one location (-) to one of two other locations (- -); The blocks are designated by the numerical values 3, 2, and 1, which represent the magnitude of the block:

```
1                                                       1
2       2                   1       1           2       2
3       3 1     3 1 2   3   2     3 2   1 3 2   1 3       3
- - -   - - -   - - -   - - -   - - -   - - -   - - -   - - -
```

Example:

```
 INPUT: Enter N: 3
OUTPUT: 7
```

3.10 Write a program that will find the set of numbers P,Q,R, where the following conditions hold:

1. Q is a 2-digit number and R is a 3-digit number.
2. P is a 5-digit number that is the product of Q and R.
3. Each of the digits 0-9 is found exactly once in the number sentence P=QxR.
4. Q is as small as possible but greater than S, where S is a 2-digit number input.

Examples:

INPUT: Enter S: **44**

OUTPUT: **P = 17820 Q = 45 R = 396**

INPUT: Enter S: **52**

OUTPUT: **P = 16038 Q = 54 R = 297**

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '86

1.1 Write a program that will first clear the screen and then center on the screen the sentence: **THIS IS THE EASIEST PROGRAM!**

1.2 Write a program that will accept two integers between -170 and 170, inclusive, and will print out the sum, difference (second from first), and product, each clearly labeled. Note: one or two spaces may separate the equal sign from each number output. Examples:

INPUT: Enter two numbers: **6, 8**

OUTPUT: **SUM = 14**
DIFFERENCE = -2
PRODUCT = 48

1.3 Consider the series

$$1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + \ldots$$

where there are infinitely many terms. Write a program to compute the sum of the series until the difference between the present sum and the sum obtained by including just one more term is less than a test value, E, given by the operator. Output must be printed in the form #.######. Example:

INPUT: Enter test value E: **0.01**

OUTPUT: **1.287037**

1.4 Write a program for Ben's Towing Service for the purpose of writing checks. The payee's first name, middle name or initial, last name, and amount will be entered separately by the operator. The program should then print a check surrounded by a border of asterisks (39 x 11) with Ben's business address at the top left, as shown below. The middle name must be abbreviated to an initial whether the operator enters the full middle name or just an initial. The middle initial must have a period immediately following. The first and last names combined will not exceed 10 letters. The amount entered must appear immediately after a dollar sign. Example:

INPUT: Enter first name: **NORM**
Enter middle name: **N**
Enter last name: **SAND**
Enter amount: **1234.56**

OUTPUT: (appears on next page)

```
OUTPUT: ****************************************
        *                                      *
        * BEN'S TOWING SERVICE                 *
        * 4563 WRECKER AVENUE                  *
        * WAVERLY, ARKANSAS 45632              *
        *                                      *
        *  PAY TO THE ORDER OF NORM N. SAND    *
        *                                      *
        *  THE SUM OF $1,234.56                *
        *                                      *
        ****************************************
```

1.5 A jail has 100 cells. In celebration of the warden's birthday, some of the prisoners will be released today. First the warden opens all the cell doors. Then starting with the first door, he closes every other door. Then, starting with the first door, he goes to every third door, closing the open doors and opening the closed ones; That is, doors 1, 4, 7, etc. He repeats this process of opening closed doors and closing open doors for doors 1,5,9, etc.; i.e. for every 4th door starting with the first door. He continues doing this for every 5th door, every 6th door, etc., always starting with the first door, until he has done this for every 100th door. Prisoners whose door is open when he is done are permitted to leave. Write a program that determines who can leave, and display each cell number in the format shown below:

```
OUTPUT: CELL 2
        CELL 5
        CELL ###
           :
           :
        CELL ###
```

1.6 Jill is thinking about opening a retirement account. Write a program to help her figure how much she can accumulate. The operator should be asked to input the monthly investment, the end of the year lump sum investment and the rate of interest. Assume that the monthly investment is deposited on the first of each month and that interest is compounded at the end of each month. Also assume that the yearly investment is deposited at the end of each year after the month's interest has been added. Calculate the total amount (rounded to the nearest penny) in the account at the end of twenty years. Example:

```
 INPUT: Enter Monthly investment: 100
        Enter end of year deposit: 600
        Enter annual rate of interest: 12

OUTPUT: AMOUNT AT END OF YEAR 20 IS $146715.74
```

1.7 Write a program which will teach your computer to print with a Southern accent. For purposes of this program, this means that for any sentence which is input (without a period), the last "g" of any word ending with ing or with ings must be dropped. Example:

INPUT: Enter sentence: **THOSE EARRINGS ARE TO MY LIKING**
OUTPUT: **THOSE EARRINS ARE TO MY LIKIN**

1.8 Assume that rabbits reproduce at the rate of 20% per month until the end of the month that overpopulation occurs, at which time they begin dying at the rate of 15% per month. Rabbits continue to die at this rate until their population is reduced by one third of the amount in which over population occurs, at which time they begin reproducing again. Write a program to simulate the rabbit population for 23 months, displaying each amount rounded to the nearest whole number. The program will accept as input the initial population and the number at which overpopulation occurs. Example:

INPUT: Enter initial population: **1826**
Enter point of over population: **2600**

OUTPUT: **POPULATION FOR MONTH 1 IS 2191**
POPULATION FOR MONTH 2 IS 2629
POPULATION FOR MONTH 3 IS 2235
POPULATION FOR MONTH 4 IS 1900
POPULATION FOR MONTH 5 IS 1615
POPULATION FOR MONTH 6 IS 1938
POPULATION FOR MONTH 7 IS 2325
POPULATION FOR MONTH 8 IS 2790
POPULATION FOR MONTH 9 IS 2372
POPULATION FOR MONTH 10 IS 2016
POPULATION FOR MONTH 11 IS 1714
POPULATION FOR MONTH 12 IS 2056
POPULATION FOR MONTH 13 IS 2468
POPULATION FOR MONTH 14 IS 2961
POPULATION FOR MONTH 15 IS 2517
POPULATION FOR MONTH 16 IS 2139
POPULATION FOR MONTH 17 IS 1819
POPULATION FOR MONTH 18 IS 2182
POPULATION FOR MONTH 19 IS 2619
POPULATION FOR MONTH 20 IS 2226
POPULATION FOR MONTH 21 IS 1892
POPULATION FOR MONTH 22 IS 1608
POPULATION FOR MONTH 23 IS 1930

1.9 The inhabitants of Eeland write normally with the exception of the fact that the letter E is always doubled when it is used. Write a program which will convert a normally written English sentence into an Eeland sentence. Example:

INPUT: Enter sentence: **SHE WORKS LIKE A BEE.**
OUTPUT: **SHEE WORKS LIKEE A BEE.**

1.10 Write a program which will accept a list of 12 whole numbers and a list of 11 whole numbers by entering one number at a time. The program will then print out the common elements of the two lists without repetition, with each number separated by 2 spaces and in the order that they appear in the first list. Example:

```
 INPUT: Enter 1 of 12: 5
        Enter 2 of 12: 6
        Enter 3 of 12: 7
        Enter 4 of 12: 9
        Enter 5 of 12: 8
        Enter 6 of 12: 9
        Enter 7 of 12: 0
        Enter 8 of 12: 23
        Enter 9 of 12: 456
        Enter 10 of 12: 789
        Enter 11 of 12: 1
        Enter 12 of 12: 12

        Enter 1 of 11: 1
        Enter 2 of 11: 23
        Enter 3 of 11: 32
        Enter 4 of 11: 23
        Enter 5 of 11: 8
        Enter 6 of 11: 99
        Enter 7 of 11: 9
        Enter 8 of 11: 10
        Enter 9 of 11: 12
        Enter 10 of 11: 8
        Enter 11 of 11: 1

OUTPUT: 9  8  23  1  12
```

2.1 Write a program that takes a sentence and right-justifies it on a line that has 65 columns. When the sentence is right-justified, it begins in column 1 and ends in column 65. The spacing between words should be approximately uniform. Example:

INPUT: Enter sentence: **THIS IS REALLY A SHORT SENTENCE.**

OUTPUT: (The sentence is to be right-justified on a 65 column line. Spacing between words is approximately uniform.)
THIS IS REALLY A SHORT SENTENCE.

2.2 Write a program that produces the following repeating pattern:

```
XXX--XXX--XXX--XXX--
---XX---XX---XX---XX
XXX--XXX--XXX--XXX--
---XX---XX---XX---XX
```

Note that on each line the X and - pattern is repeated four times. Also, the second line has -'s where the first line has X's and it has X's where the first line has -'s. The third and fourth lines repeat the pattern in the first two lines. Write the program so that the operator puts in the total number of X's and -'s used in the basic sequence, the number of X's used in the beginning basic sequence, and the number of rows to be printed (less than 20). Example:

INPUT: Enter total number of X's and -'s: **5**
Enter number of X's: **3**
Enter number of rows: **5**

OUTPUT:
```
XXX--XXX--XXX--XXX--
---XX---XX---XX---XX
XXX--XXX--XXX--XXX--
---XX---XX---XX---XX
XXX--XXX--XXX--XXX--
```

2.3 You just got a job with a secret agency as a message coder/decoder. Write a program which gives the operator the choice of: 1) coding a message from English into a secret code; or 2) decoding a message from secret code into English. The master translation string is "ZXCVBNMASDFGHJKLQWERTYUIOP " for each of the corresponding letters of the alphabet. Notice that the space is the last character of the translation string which always translated to a space. Only these characters will be used. The program is to display a menu of three choices as shown below and will continue to prompt for input until a '3' is entered. Example:

```
OUTPUT: 1) ENCODE
        2) DECODE
        3) END
 INPUT: Choose: 1
        Enter message: HERE IS A MESSAGE
OUTPUT: ABWB SE Z HBEEZMB

        1) ENCODE
        2) DECODE
        3) END
 INPUT: Choose: 2
        Enter message: ABWB SE Z HBEEZMB
OUTPUT: HERE IS A MESSAGE

        1) ENCODE
        2) DECODE
        3) END
 INPUT: Choose: 3
OUTPUT: (program terminates)
```

2.4 The mode of a set of numbers is the most frequently used number. Write a program in which the operator is asked to input 15 numbers. The program then finds and prints the unique mode if only one mode; otherwise have the computer print the message NO UNIQUE MODE. Examples:

```
 INPUT: Enter number 1: 7
        Enter number 2: 1
        Enter number 3: 4
        Enter number 4: 9
        Enter number 5: 5
        Enter number 6: 4
        Enter number 7: 5
        Enter number 8: 7
        Enter number 9: 1
        Enter number 10: 5
        Enter number 11: 22
        Enter number 12: 33
        Enter number 13: 44
        Enter number 14: 55
        Enter number 15: 4

OUTPUT: NO UNIQUE MODE
```

```
 INPUT: Enter number 1: 1
        Enter number 2: 2
        Enter number 3: 3
        Enter number 4: 4
        Enter number 5: 5
        Enter number 6: 6
        Enter number 7: 7
        Enter number 8: 8
        Enter number 9: 5
        Enter number 10: 1
        Enter number 11: 2
        Enter number 12: 3
        Enter number 13: 4
        Enter number 14: 5
        Enter number 15: 6

OUTPUT: MODE IS 5
```

2.5 A bank needs a program to handle savings accounts. The original balance in an account will be entered at the terminal. The annual interest rate on deposits will be 7%. The rest of the program will be menu driven. Choices will be: 1. MAKE A DEPOSIT, 2. MAKE A WITHDRAWAL, 3. CREDIT INTEREST, or 4. END the program. The amount of a deposit or a withdrawal will be entered at the terminal. After each transaction, the program must print the balance before the transaction, the type of transaction (deposit, withdrawal, or interest) with the amount of the transaction, and the new balance after the transaction (always above $1500). When crediting interest, the current balance is the amount on which to figure the month's interest. When the "end" choice is selected, the final balance must be printed before terminating. Example:

```
 INPUT: Enter original balance: 2345.15
OUTPUT: 1. MAKE A DEPOSIT
        2. MAKE A WITHDRAWAL
        3. CREDIT INTEREST
        4. END
 INPUT: Enter option: 1

        Enter amount to deposit: 100
OUTPUT: BALANCE BEFORE TRANSACTION $2,345.15
        MAKE A DEPOSIT
        NEW BALANCE $2,445.15

        1. MAKE A DEPOSIT
        2. MAKE A WITHDRAWAL
        3. CREDIT INTEREST
        4. END
 INPUT: Enter option: 2

        Enter amount to withdraw: 50
OUTPUT: BALANCE BEFORE TRANSACTION $2,445.15
        MAKE A WITHDRAWAL
        NEW BALANCE $2,395.15

        1. MAKE A DEPOSIT
        2. MAKE A WITHDRAWAL
        3. CREDIT INTEREST
        4. END
 INPUT: Enter option: 3

OUTPUT: BALANCE BEFORE TRANSACTION $2,395.15
        CREDIT INTEREST OF $ 13.97
        NEW BALANCE $2,409.12

        1. MAKE A DEPOSIT
        2. MAKE A WITHDRAWAL
        3. CREDIT INTEREST
        4. END
 INPUT: Enter option: 4

OUTPUT: FINAL BALANCE $3,409.12
```

2.6 Write a program in which the operator may input any two positive integers (with at most 38 digits each), and the program then prints the sum of these two numbers. Example:

```
INPUT: Enter first number:  54387698325431256754
       Enter second number: 76589787321212546786

OUTPUT: SUM IS 130977485646643803540
```

2.7 Write a menu driven program which can perform decimal/metric conversions in each direction with the choices below. Each of the 12 choices should allow an amount to be entered and then print the equivalent amount in the other system of measure. Each amount displayed must be in the form ###.###. A 13th choice will end the program. The menu must be presented in two columns with six choices in the first column and the other six in the second column (with the two-digit numbers beginning in column 27) as shown below. Note:

```
1 inch   = 2.54 centimeters
1 foot   = 0.3048 meters
1 mile   = 1.6093 kilometers
1 ounce  = 28.35 grams
1 pound  = 0.4536 kilograms
1 gallon = 3.7854 liters
```

Example:

```
OUTPUT:  1 CENTIMETERS TO INCHES   7 GRAMS TO OUNCES
         2 INCHES TO CENTIMETERS   8 OUNCES TO GRAMS
         3 METERS TO FEET          9 KILOGRAMS TO POUNDS
         4 FEET TO METERS         10 POUNDS TO KILOGRAMS
         5 KILOMETERS TO MILES    11 LITERS TO GALLONS
         6 MILES TO KILOMETERS    12 GALLONS TO LITERS
                                  13 END
 INPUT: Enter option: 2
        Enter number of INCHES: 100

OUTPUT: THIS IS EQUIVALENT TO 254.000 CENTIMETERS

OUTPUT:  1 CENTIMETERS TO INCHES   7 GRAMS TO OUNCES
         2 INCHES TO CENTIMETERS   8 OUNCES TO GRAMS
         3 METERS TO FEET          9 KILOGRAMS TO POUNDS
         4 FEET TO METERS         10 POUNDS TO KILOGRAMS
         5 KILOMETERS TO MILES    11 LITERS TO GALLONS
         6 MILES TO KILOMETERS    12 GALLONS TO LITERS
                                  13 END
 INPUT: Enter option: 10
        Enter number of POUNDS: 10

OUTPUT: THIS IS EQUIVALENT TO   4.536 KILOGRAMS
OUTPUT: (menu displays again)
 INPUT: Enter option: 13

OUTPUT: (program terminates)
```

2.8 Write a program to generate a mortgage amortization where the user inputs the principal, the interest rate, the term in years, and the number of the month during which the first payment is made. Have the program print the amount of interest paid for each month and the outstanding principal for that month. At the end of each year the program is to print the total amount of interest paid during that year and pause until a key has been pressed so that all the output may be viewed on the screen before proceeding to the next set of amounts. At the end of the loan, have the program print the total interest paid and the calculated monthly payment. Each amount must be rounded to the nearest penny and displayed in the form ###.## for monthly interest and #####.## for all others. Note: The formula for monthly payment is:

```
PA = (R*(1+R)^(Y*12))/((1+R)^(12*Y)-1)*P

      where R is decimal interest,
            Y is term in years,
        and P is principal.
```

Note: The program will not compute accurate amounts if the power symbol (^) is used. The final principal must be -0.00 or 0.00.

A partial listing of a sample output is given below:

```
 INPUT: Enter principal: 10000
        Enter % rate of interest: 10
        Enter term in years: 4
        Enter # of month in year for first payment: 4
OUTPUT: INTEREST          PRINCIPAL
        $ 83.33           $ 9829.71
        $ 81.91           $ 9658.00
        $ 80.48           $ 9484.85
        $ 79.04           $ 9310.27
        $ 77.59           $ 9134.23
        $ 76.12           $ 8956.72
        $ 74.64           $ 8777.73
        $ 73.15           $ 8597.26
        $ 71.64           $ 8415.27

        YEAR'S INTEREST   $  697.91

        $ 70.13           $ 8231.78
                  :
                  :
        YEAR'S INTEREST   $  229.58

        $  6.24           $  500.98
        $  4.17           $  251.53
        $  2.10           $   -0.00

        YEAR'S INTEREST   $   12.51
        TOTAL INTEREST    $ 2174.04
        MONTHLY PAYMENT   $  253.63
```

2.9 The value of sine x is given by the series

x - x^3/3! + x^5/5! - x^7/7! + x^9/9! - x^11/11! + ...

where x is measured in radians and the series has an infinite number of terms. Write a program that calculates the sine of N degrees, once N is entered as a number greater than 0 and less than 360. Perform the calculation so that x is added to -x^3/3! and the sum is added to x^5/5! and so on. The last term to be added will be -x^11/11!. Have the computer print the sum followed by the actual value of sine of N degrees as calculated by the computer's internal sine function. Each value is to be displayed in the form #.#######.

Note: Using the power sign (^) could give an inaccurate result.
Note: When N is greater than 180 use the reference angle of 360 - N for accurate results, but be careful with the sign.
Note: PI is approximately equivalent to 3.1415926535.

Examples:

INPUT: Enter N degrees: **125**

OUTPUT: **PARTIAL SUM = 0.8191481**
ACTUAL SINE = 0.8191520

INPUT: Enter N degrees: **270**

OUTPUT: **PARTIAL SUM = -0.9999999**
ACTUAL SINE = -1.0000000

2.10 The seven Roman numerals and their corresponding values in the Arabic system are M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5, and I = 1. The Roman system is generally the sum of the values of the digits as long as the digits decrease to the right. That is, MDLXXVII would be 1577.Whenever a digit of smaller value appears to the left, it is subtracted from the digit to its immediate right. Only C, X, and I may subtract. C will subtract from M or D, X from C or L, and I from X or V. The number MCDXLIX is 1449. Write a program to accept as input a Roman numeral and then translate the Roman numeral to an Arabic numeral. Examples:

INPUT: Enter Roman Numeral: **MMCMXCIX**
OUTPUT: **ARABIC = 2999**

INPUT: Enter Roman Numeral: **CCCXXXVIII**
OUTPUT: **ARABIC = 338**

3.1 Write a program that produces monthly calendars for the year 1986, one month at a time, starting with January. For each calendar the name of the month must be approximately centered above the calendar days. An abbreviation of the names of each day (starting with Sunday) must be on the top of each column of day numbers. Only the month of January needs the heading '1986'. After one monthly calendar is displayed, allow the user to press any key to clear the screen and to display the next month. Each month should have a neat form similar to the calendar of January as shown below.

Hint: remember the rhyme, 30 days have September, April, June, and November. All the rest have 31 except February which has 28 and on a leap year 29. Partial listing of example:

1986

JANUARY

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

FEBRUARY

S	M	T	W	T	F	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

3.2 A fundamental theorem of mathematics says that every 5th degree polynomial has at least one root (a value of the variable at which the polynomial equals 0).

Write a program which will find a root for any 5th degree polynomial in the variable x (whose root lies between -10 and 10), once given the values of the coefficients of the polynomial:

$$Ax^5 + Bx^4 + Cx^3 + Dx^2 + Ex + F = 0$$

The root value x must be displayed in the form #.##### with the 5th digit on the right of the decimal being rounded. Example:

INPUT: Enter coefficients A,B,C,D,E,F: **1, -5, 8, 5, -4, 3**
OUTPUT: **ROOT IS -0.93629**

3.3 Write a program which changes a base A numeral into a base B numeral, where A and B are any integers between 2 and 36, inclusive. Examples:

```
INPUT: Enter base A: 12
       Enter base B: 11
       Enter original number: A14B

OUTPUT: A14B BASE 12 EQUALS 12154 BASE 11


INPUT: Enter base A: 36
       Enter base B: 27
       Enter original number: 1Y

OUTPUT: 1Y BASE 36 EQUALS 2G BASE 27
```

3.4 A number of DATA statements contain customer accounts. For each customer, there are four fields: social security number in the form XXXXXXXXX, name, address, and account balance. Store the customer data into associated arrays. Additional input will come from the terminal and will consist of three items: social security number, C for charge or P for payment, and amount of transaction. The social security number is used to determine whose balance is affected; the balance is then updated; and the new balance is printed. The process is repeated until the social security number 000000000 is entered. After that, all data is to be printed in a report showing social security number, name, address, and the final balance for each customer. The customer listing must be arranged in decreasing order according to the account balance. Balances are displayed in the form ####.##. Use the DATA statements given below.

```
DATA 234567890,JOHN SMITH,
     "1234 ANYWHERE LANE, EXIST, KANSAS 66754",345.78
DATA 564783219,GAIL HUSTON,
     "543 SOUTH THIRD, BIG TOWN, TEXAS 88642",2365.89
DATA 873421765,TIM JONES,
     "2387 PALM PLACE, NOME, ALASKA 77643",6754.76
DATA 543876543,JILL RUPERTS,
     "4536 123RD STREET, TINY TOWN, MAINE 76765",45.18
DATA 345212342,AL BROWN,
     "PO BOX 234, TINSEL TOWN, CALIFORNIA 77654",3456.09
DATA 565656565,KERMIT TEU,
     "1234 LOST LANE, WIMPLE, WISCONSIN 66543",78.36
```

Example:

```
INPUT: Enter SSN: 564783219
       Enter C for charge or P for payment: C
       Enter amount of transaction: 10

OUTPUT: NEW BALANCE IS $2355.89
```

(Input/Output continued on next page)

(Input/Output continued)

```
 INPUT: Enter SSN: 565656565
        Enter C for charge or P for payment: P
        Enter amount of transaction: 10

OUTPUT: NEW BALANCE IS $  88.36

 INPUT: Enter SSN: 000000000

OUTPUT: SSN        NAME          ADDRESS              BALANCE

        873421765  TIM JONES     2387 PALM PLACE      $6754.76
                                 NOME
                                 ALASKA 77643
        345212342  AL BROWN      PO BOX 234           $3456.09
                                 TINSEL TOWN
                                 CALIFORNIA 77654
        564783219  GAIL HUSTON   543 SOUTH THIRD      $2355.89
                                 BIG TOWN
                                 TEXAS 88642
        234567890  JOHN SMITH    1234 ANYWHERE LANE   $ 345.78
                                 EXIST
                                 KANSAS 66754
        565656565  KERMIT TEU    1234 LOST LANE       $  88.36
                                 WIMPLE
                                 WISCONSIN 66543
        543876543  JILL RUPERTS  4536 123RD STREET    $  45.18
                                 TINY TOWN
                                 MAINE 76765
```

3.5 Write a program to print out the product of any two decimal numbers, each containing at most 20 digits and a decimal point. Example:

```
 INPUT: Enter first number: 1.123456789
        Enter second number: 0.11239

OUTPUT: PRODUCT = 0.12626530851571


 INPUT: Enter first number: 123456789.123456789
        Enter second number: 1000000000.0000000001

OUTPUT: PRODUCT = 123456789123456789.123456789123456789
```

3.6 A palindrome is a word, verse, or number that reads the same backwards or forwards. The number 12321 is a palindrome. Here is a method of forming palindromic numbers which is almost always successful: Begin with any positive integer. If it is not a palindrome, reverse its digits and add the two numbers. If the sum is not a palindrome, treat it as the original number and continue until a palindrome is generated. Write a program to find palindromes using this method; However, allow a limit of at most 23 additions. For example, 196 will not produce a palindrome, so a limit is necessary. If a number will not produce a palindrome after 23 additions, then print out the message: CANNOT GENERATE A PALINDROME. Note: the generated palindrome could be up to 15 digits long. In the example below, 5405 can generate a palindrome because 5405 + 5045 = 10450 and 10450 + 5401 = 15851.

Example:

INPUT: Enter number: **5405**
OUTPUT: **15851 IS A PALINDROME**

INPUT: Enter number: **196**
OUTPUT: **CANNOT GENERATE A PALINDROME**

3.7 Write a program to solve a system of N equations with N unknowns. The program will first accept a value for N between 2 and 4 inclusive. For each equation the program should accept the values for the N coefficients followed by the constant. Each given system of equations will have one unique solution whose values will be integers. A possible algorithm to be used to solve an NxN system of equations is shown to the right of the example input/output. Examples:

```
INPUT: Enter N: 3
Enter coefficients for row1
Co1: 2
Co2: 4
Co3: 6
Enter constant: 4
Enter coefficients for row2
Co1: 2
Co2: 1
Co3: 1
Enter constant: 3
Enter coefficients for row3
Co1: -1
Co2: 2
Co3: 1
Enter constant: 2

OUTPUT: (1, 2, -1)
```

```
 2x + 4y + 6z = 4    (Divide by 2)
 2x +  y +  z = 3
-x + 2y +  z = 2

 1  2   3   2
 2  1   1   3   (Subtract 2*Row1)
-1  2   1   2

 1  2   3   2
 0 -3  -5  -1
-1  2   1   2   (Subtract -1*Row1)

 1  2   3   2
 0 -3  -5  -1   (Divide by -3)
 0  4   4   4

 1  2   3   2
 0  1  5/3 1/3
 0  4   4   4   (Subtract 4*Row2)

 1  2   3   2
 0  1  5/3 1/3
 0  0 -8/3 8/3  (Divide by -8/3)

 1  2   3   2
 0  1  5/3 1/3  (Subtract 5/3*Row3)
 0  0   1  -1

 1  2   3   2   (Subtract 3*Row3)
 0  1   0   2
 0  0   1  -1

 1  2   0   5   (Subtract 2*Row2)
 0  1   0   2
 0  0   1  -1

 1  0   0   1
 0  1   0   2
 0  0   1  -1
```

```
INPUT: Enter N: 2
Enter coefficients for row1
Co1: 1
Co2: 1
Enter constant: 1
Enter coefficients for row2
Co1: 2
Co2: 3
Enter constant: 6

OUTPUT: (-3, 4)
```

3.8 Write an EFFICIENT program to accept as input a word with L distinct letters (L is less than 8) and a positive integer K. Your program is to print out the Kth, 2*Kth, and 3*Kth elements of the alphabetized list of all the permutations of the word. Each permutation is to be separated by two spaces. For example, if the word CAT is entered with K=2 then the computer would form the list ACT, ATC, CAT, CTA, TAC, TCA, and output would be: ATC CTA TCA. NOTE: No team completed this program last year. Can anyone do it this year!

Example:

```
 INPUT: Enter word: CAT
OUTPUT: ATC  CTA  TCA
```

3.9 Write an EFFICIENT program to solve the following cryptorithm puzzle where each unique letter represents a unique digit:

ABB - CB = DEF

CB is a 2-digit number and is subtracted from the 3-digit number ABB to obtain the 3-digit result DEF. Have the program print every unique solution, with each one numbered, and the total number of solutions printed at the end. The solutions do not have to appear in any particular order, but they must appear in the format below. A partial listing of an example:

```
OUTPUT: 411 - 21 = 390  NUMBER 1
        511 - 21 = 490  NUMBER 2
        611 - 21 = 590  NUMBER 3
        711 - 21 = 690  NUMBER 4
        :
        :
        :
        477 - 97 = 380  NUMBER ###
        577 - 97 = 480  NUMBER ###
        677 - 97 = 580  NUMBER ###

        TOTAL NUMBER OF SOLUTIONS = ###
```

3.10 Write a program that will find all positive two digit integers that can exist as the sum of the digits 0-9 in which each of the digits are used exactly once. Any two of the digits may be combined to form a 2-digit integer to be added with the other digits. Output must include every qualified number in ascending order followed by ONE OF THE SUM PROCESSES that qualifies the number. Order of the addends does not matter. A partial example:

```
OUTPUT: 45 = 0 + 1 + 2 + 3 + 4 + 5 + 7 + 8 + 9
        :
        72 = 10 + 23 + 4 + 5 + 6 + 7 + 8 + 9
        :
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '87

1.1 Write a program to accept a number and print out the value of its sign: POSITIVE, NEGATIVE, or ZERO.

```
 INPUT: Enter number: -4.5
OUTPUT: NEGATIVE
```

1.2 Write a program that accepts an integer, n, as input and then finds and prints the sum of the numbers n, n+1, n+2, ..., n+20. Example:

```
 INPUT: Enter n: 3

OUTPUT: SUM = 273
```

1.3 Write a program to print PROBLEM THREE diagonally down and across the screen. The output must be centered on the screen, both from top to bottom and left to right. Example:

```
P
 R
  O
   B
    L
     E
      M

        T
         H
          R
           E
            E
```

1.4 A standard six sided die has a 1 on the opposite side of the 6, a 3 on the opposite side of the 4, and a 5 on the opposite side of the 2. Write a program that accepts input for the numbers on the top, front, and right sides of the die and then prints the numbers on each of the six sides of the die. Example:

```
INPUT: Enter number on top: 1
       Enter number on front: 3
       Enter number on right: 5

OUTPUT: TOP= 1
        FRONT= 3
        RIGHT= 5
        BOTTOM= 6
        BACK= 4
        LEFT= 2
```

1.5 Write a program to fill the entire screen with random characters, then pause. Upon pressing any key, the screen will clear.

1.6 Write a program to accept two sets of two integers which correspond to the upper left and lower right hand coordinates of a rectangle on the screen. Fill this imaginary rectangle with periods. Everything else on your screen should be blank. In the example below, the first period is to appear in the fourth position of line four on the screen. Example:

```
 INPUT: Enter coordinates: 4,4, 7,12

OUTPUT: .........
        .........
        .........
        .........
```

1.7 An easy and efficient method of generating random numbers is to start with an initial value, the seed, multiply by 421, add 1 to the product, then divide by 100 and use the remainder as the random number, and as the new seed. Write a program which accepts a value for the seed and prints out the next 10 random numbers. Example:

```
 INPUT: Enter seed: 3

OUTPUT: 64
        45
        46
        67
        8
        69
        50
        51
        72
        13
```

1.8 Write a program to determine the mass of a big fish tank that is filled to the top with water if the tank (without water) has a mass of K kilograms and has dimensions of L feet by W feet by H feet. K,L,W,H will be input. Output will be the mass in the form #####.## KILOGRAMS.

Note: 1 inch = 2.54 centimeters;
1 cubic cm. of water = 1 gram

Example:

```
 INPUT: Enter K, L, W, H: 32, 5, 4, 2
OUTPUT:  1164.67 KILOGRAMS
```

1.9 Write a program to clear the screen and display the design shown below, exactly. It consists of 21 rows made up of the letters in the alphabet. There are 31 letters in every odd numbered row. In every even numbered row there are 11 columns of letters separated by 2 spaces. Example:

OUTPUT:
```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
B  B  B  B  B  B  B  B  B  B  B
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
D  D  D  D  D  D  D  D  D  D  D
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
F  F  F  F  F  F  F  F  F  F  F
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
H  H  H  H  H  H  H  H  H  H  H
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
J  J  J  J  J  J  J  J  J  J  J
KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
L  L  L  L  L  L  L  L  L  L  L
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
N  N  N  N  N  N  N  N  N  N  N
OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
P  P  P  P  P  P  P  P  P  P  P
QQQQQQQQQQQQQQQQQQQQQQQQQQQQQQQ
R  R  R  R  R  R  R  R  R  R  R
SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS
T  T  T  T  T  T  T  T  T  T  T
UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
```

1.10 Write a program to print the number of hours and minutes (clearly labeled) a reader takes to read a given book if he reads at a given rate (minutes/page). Both the book title and the rate will be input. The minutes displayed must be between 0 and 59, inclusive. The program will use the contents of the four DATA lines shown below in some manner. Each DATA line consists of the title of a book and the number of pages in the book, as shown below.

```
DATA THE HISTORY OF THE COMPUTER, 400
DATA THE RED DOG RUNS, 200
DATA EATING APPLE PIE, 150
DATA THE ART OF WINNING, 250
```

Example:

INPUT: Enter book title: **EATING APPLE PIE**
Enter rate (minutes/page): **1**

OUTPUT: **2 HOURS 30 MINUTES**

2.1 Write a program to accept a string of up to ten letters, and an integer, N. First print the string. Then on the next line, print the string rotated to the left N times. When a character "falls off" the left end, it must be wrapped onto the right end. Example:

```
 INPUT: Enter string: ABCDEFG
        Enter N: 3

OUTPUT: ABCDEFG
        DEFGABC
```

2.2 A computer salesperson buys stock in large quantities. The salesperson bought a certain amount of cartons of each of 3 types of diskettes. Each carton contains 100 diskettes. Vers cost $225 per carton, Maxs cost $297 per carton, and Wabs cost only $120 per carton. If a total of 100 cartons were bought, how many cartons of each type makes the total bill come to $23,607? Display output in the form below. Example:

```
OUTPUT: ## VERS  ## MAX  ## WABS
```

2.3 Using the random-number generator, write a program that accepts a list of at most 15 unique positive integers (ending list with a -1) and outputs a random set of five of those integers each time a key is pressed. The random list should differ each time the key is pressed, and should contain no duplicates. Example:

```
 INPUT: Enter list item: 3
        Enter list item: 4
        Enter list item: 76
        Enter list item: 31
        Enter list item: 47
        Enter list item: 88
        Enter list item: 1
        Enter list item: 5
        Enter list item: 901
        Enter list item: 23
        Enter list item: 7
        Enter list item: -1
OUTPUT: (Possible output, but will differ)
        901
        3
        76
        88
        23
        PRESS ANY KEY
 INPUT: (Press any key)
OUTPUT: 47
        1
        31
        5
        76
```

2.4 Write a program to produce a complete listing of all the ways a given positive integer less than 20 can be partitioned as a sum of equal numbers. The listing must be in pyramid form as shown below with only the + between the addends (no spaces). Example:

INPUT: Enter a number less than 20: **12**

OUTPUT:
```
                12
               6+6
              4+4+4
             3+3+3+3
           2+2+2+2+2+2
      1+1+1+1+1+1+1+1+1+1+1+1
```

2.5 Write a program to calculate the fractional value of three letter words. The value of a letter in the alphabet (A...Z) is defined as the position of that letter in the alphabet. Thus A=1, B=2, C=3 ... Z=26. The fractional value is the sum of the reciprocals of the value of each letter in that word. This value must be printed out as a simplified fraction. In the example below, 1/3 + 1/1 + 1/2 = 11/6. Example:

INPUT: Enter word: **CAB**

OUTPUT: **11/6**

2.6 Write a program that accepts a set of at most 7 integers, an integer N, and another integer S, and determines if there is a subset of N elements from the set that sums to no more than S: output "YES" or "NO". Your output must include the elements in one of the subsets, if one exists, and display that set in ascending order. Indicate the end of your input set with a -1. Example:

INPUT: Enter set item: **6**
Enter set item: **8**
Enter set item: **2**
Enter set item: **14**
Enter set item: **3**
Enter set item: **-1**
Enter N: **3**
Enter S: **15**

OUTPUT: **YES**
2 3 6
(other subsets may be displayed instead, e.g. 2,3,8)

2.7 Write a program that recognizes strings of the following pattern type: an "A" followed by zero or more "BA"s followed by one or more "A"s. Some legal patterns are AA, ABAA, ABABAAAAA, and so on. Illegal patterns include AABAA, ABABABA. Example:

```
 INPUT: Enter pattern: BABAAA
OUTPUT: ILLEGAL PATTERN

 INPUT: Enter pattern: AAA
OUTPUT: LEGAL PATTERN
```

2.8 Write an efficient program to find and print all integers between M and N inclusive that have exactly F distinct positive factors. M,N, and F will be inputted with M greater than 1, M less than N, N less than 1000, and F greater than 1. Example:

```
 INPUT: Enter M, N, F: 2, 10, 2

OUTPUT: 2
        3
        5
        7
```

2.9 Write a program to alphabetize 5 words according to the following rules: The word containing the lowest ranking letter (considering every letter in the word) is first, then the word containing the second lowest ranking letter is second, and so on. If two words have the same lowest ranking letter, then compare each word's next lowest ranking letter, and so on. If during the process a word runs out of letters, then it is the next word printed. For example:

```
TAP comes before PAY because...
Both have the same lowest ranking letter, A.
Both have the same second lowest ranking letter, P.
But, TAP's T comes before PAY's Y.
```

Example:

```
 INPUT: Enter word 1: FLORIDA
        Enter word 2: HIGH
        Enter word 3: SCHOOLS
        Enter word 4: COMPUTING
        Enter word 5: COMPETITION

OUTPUT: FLORIDA
        COMPETITION
        COMPUTING
        SCHOOLS
        HIGH
```

2.10 Write a program to produce a super-duper input routine with 4 types of input. The program should accept the ROW position and COLumn position of the first character to be input. The program should also accept a number for the MAXimum number of characters allowed for input. Also, accept the TYPE of input (1-4). The input routine must not allow more than MAX characters to be entered. By pressing a BACKSPACE key the program will remove the last character printed and wait for a new character to replace it. The program must not allow the user to backspace past the original ROW and COLumn positions entered. According to the TYPE of entry (1,2,3, or 4), the following restrictions are placed upon the characters being entered:

```
TYPE 1: Only Alphabetic letters and a space allowed.
TYPE 2: Only Numeric digits and decimal points allowed.
TYPE 3: Only Numeric digits and dashes [ - ] allowed in
        the following date format POSITIONS:  MM-DD-YY.
        MAX will be entered as 8 for this entry.
TYPE 4: All characters are allowable.
```

When the RETURN key is pressed, the program will display the entry two rows directly beneath the characters that were typed on the screen. Example:

```
 INPUT: Enter ROW, COL: 5, 5
        Enter MAX: 8
        Enter TYPE: 1
        (The program should accept input at ROW 5, COLumn 5)

 INPUT: ABCD F2
OUTPUT: ABCD F      (The program must not display the "2")
 INPUT: GHI
OUTPUT: ABCD FGH
 INPUT: (Press the BACKSPACE key 6 times.)
OUTPUT: AB
 INPUT: -    (dash)
OUTPUT: AB
 INPUT: (RETURN key)
OUTPUT: AB   (will be printed two rows beneath the typed AB).
```

3.1 The following problem is part of an algorithm to correct misspellings. Two words are said to be close if they are the same or one word can be obtained from the other by a single transformation of the following types:

One letter is changed
One letter is deleted
One letter is added
Two adjacent letters are transposed.

Write a program to accept two words and determine if they are close. Examples:

```
 INPUT: Enter word 1: THEIR
        Enter word 2: THIER
OUTPUT: CLOSE

 INPUT: Enter word 1: THERE
        Enter word 2: THEIR
OUTPUT: NOT CLOSE
```

3.2 Write a program to evaluate an NxN determinant where N is input as 2, 3, or 4. In order to evaluate a 4x4 determinant, the program must choose a row or column of 4 numbers and add all the products of the 4 numbers with their corresponding evaluated 3x3 minor in the following way: If the numbers' row and column positions are both odd or both even then the program adds the product of that number with its 3x3 minor; otherwise, the program subtracts the product of that number with its 3x3 minor. For example, choosing the bottom row of numbers for the following 4x4 determinant:

```
 INPUT: Enter dimension N: 4
 INPUT: (enter the NxN numbers one at a time, going
        from left to right, top row to bottom row)
                      |1 2 3 4|
                      |5 6 7 8|
                      |9 0 1 2|
                      |3 4 5 6|

OUTPUT: 0             (because...)
        |2 3 4|       |1 3 4|       |1 2 4|       |1 2 3|
  - 3 x |6 7 8| + 4 x |5 7 8| - 5 x |5 6 8| + 6 x |5 6 7|
        |0 1 2|       |9 1 2|       |9 0 2|       |9 0 1|

 = -3 x (2x7x2 + 3x8x0 + 4x6x1 - 0x7x4 - 1x8x2 - 2x6x3)
   +4 x (1x7x2 + 3x8x9 + 4x5x1 - 9x7x4 - 1x8x1 - 2x5x3)
   -5 x (1x6x2 + 2x8x9 + 4x5x0 - 9x6x4 - 0x8x1 - 2x5x2)
   +6 x (1x6x1 + 2x7x9 + 3x5x0 - 9x6x3 - 0x7x1 - 1x5x2)
 = -3x(0) + 4x(-40) -5x(-80) + 6x(-40)  = 0
```

3.3 Write a program that will accept a string of text several lines long, and output a list of each word used and how many times it was used. A word is defined as any contiguous sequence of letters, possibly including an apostrophe. Legal characters are A-Z, ., (space), ', !, and ?. Display the words in the order in which they appear in the text. Example:

INPUT: Enter text: **HOW MUCH WOOD COULD A WOODCHUCK CHUCK IF A WOODCHUCK COULD CHUCK WOOD?**

OUTPUT: **1 HOW**
1 MUCH
2 WOOD
2 COULD
2 A
2 WOODCHUCK
2 CHUCK
1 IF

3.4 Write a program that accepts a string of input and encrypts it (puts it in code). Your program must be such that if the encoded string is input to the program, it will output the original string. For example, if the input is "NOW IS THE TIME" and the output is "DFEA* ASSA TESR" then if we input "DFEA* ASSA TESR" the output should be "NOW IS THE TIME". Note that there is no "switch" that tells the program whether or not the input is in code or not. Your program must be able to accept any character as inputted, including the non-printable ones (whose ASCII value is between 0 and 31 or between 93 and 255). In the case of non-printable characters, the input and output should be of the form "/nnn" where nnn is the ASCII number of the character. Thus to encode (or report) a carriage return, CHR$(13), one would enter /013. Similarly on output, a carriage return will be represented by /013. Use // to represent the character {/}. You may use /044 to enter a {,} and /058 to enter a {:} if you like. In this case you may output either the /044 or a {,}. Similarly you many output either the /058 or a {:}. The following is an example of two different runs:

INPUT: Enter text: **HERE I AM**

OUTPUT: **G6/0298F*/030[`**

INPUT: Enter text: **G6/0298F*/030[`**

OUTPUT: **HERE I AM**

HINT: You can be creative in designing an encryption routine. Your input of "HERE I AM" may give a different sequence of characters than those above. However, your sequence should give as output "HERE I AM" whenever this sequence is input.

3.5 Debbie thought up 2 four digit numbers and multiplied them together. From the product, she removed one of the digits and scrambled the rest. She also scrambled up the four digit numbers. If 5132 and 4735 were the scrambled four digit numbers and 8014153 is the scrambled product with one digit removed, write a program to find two possible sets of the 3 unscrambled numbers. Display each set with its elements in ascending order in the format shown below. The order of the two sets does not matter. Example:

```
OUTPUT: ####  ####  ########
        ####  ####  ########
```

3.6 Rubik's Pocket Cube is a two by two by two inch cube, each side containing a different color. Each side is divided into 4 square regions (dimensions of one inch by one inch). The sides of this puzzle move independently of each other. Write a program that reads into an array the color symbols (W,Y,O,G,R,B) for each of the 4 squares in its original state (4 W's on top, 4 Y's in front, 4 O's on the right, 4 G's on the back, 4 R's on the left, and 4 B's on the bottom).

Your program then repeatedly asks for one clockwise rotation of either the top side or the front side. If Q is entered instead, then the program quits. Otherwise, your program must then display the 4 color symbols that are currently on the front side (from left to right, top to bottom). Example:

```
 INPUT: Enter T, F, or Q: T
OUTPUT: O  O
        Y  Y
 INPUT: Enter T, F, or Q: F
OUTPUT: Y  O
        Y  O
 INPUT: Enter T, F or Q: Q
OUTPUT: (program terminates)
```

3.7 Write a program to simulate a mini drill and practice for adding Roman Numerals. The program first accept a user's name and the date. The program will then display the following menu after clearing the screen:

1. INSTRUCTION PAGE
2. PRACTICE 3 PROBLEMS
3. QUIT

If OPTION 3 is chosen, then the program quits.
If OPTION 1 is chosen then the following instruction page will appear after clearing the screen:

YOU WILL BE GIVEN 3 PROBLEMS TO
WORK. A PROBLEM WILL CONSIST OF
ADDING TWO RANDOMLY GENERATED
ROMAN NUMERALS LESS THAN 20.
YOU WILL TYPE YOUR ANSWER IN
ROMAN NUMERALS AND PRESS `RETURN.'
(PRESS ANY KEY TO RETURN TO MENU.)

The program then waits for a key to be pressed and then displays the menu after clearing the screen. If OPTION 2 is chosen then a problem is displayed in the center portion of the screen with each of the numerals right justified. The bottom numeral must have a + on its left with a space between. An underlined dash must be displayed on the row below the bottom numeral and extend from the + to the right-most character in the bottom numeral. Example:

```
   IV
+ XIX
-----
```

The user's response will be accepted directly under this line. If the answer is CORRECT, then the next problem is displayed. If the answer is INCORRECT, the Arabic form of the answer will be displayed somewhere close to the bottom, and the user will have one more chance to correctly answer the problem. After three problems are completed, a progress report will be displayed:

PROGRESS REPORT
DATE:
NAME:
NUMBER CORRECT:
NUMBER OF EXERCISES: 3
PERCENT CORRECT: (rounded to the nearest integer)

(If there is at least one wrong answer then the following report is also displayed with this heading and the corresponding information below the appropriate headings.)

WRONG ANSWER **CORRECT ANSWER** **ARABIC**
(user's last answer) (Roman Numeral) (The sum)

The program then displays the message "**PRESS ANY KEY TO RETURN TO MENU.**" at the bottom. When a key is pressed, the program will clear the screen and display the menu.

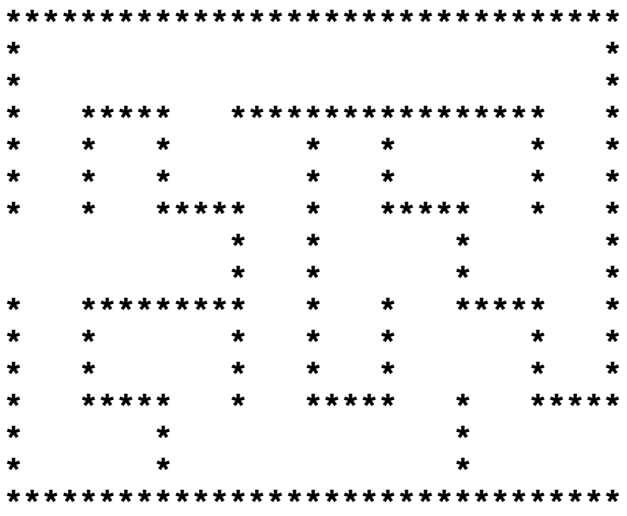
3.8 Write a program to determine the area shared in common (if any) with two rectangles given the 4 coordinates of the corners of each rectangle. The corners will have integer coordinates that lie within the third quadrant of the Cartesian plane (all coordinates will be negative and have an absolute value less than 20.) For convenience, each rectangles' coordinates will be entered beginning with the X,Y coordinate of the bottom right hand corner and go clock-wise to the other 3 corners. Example:

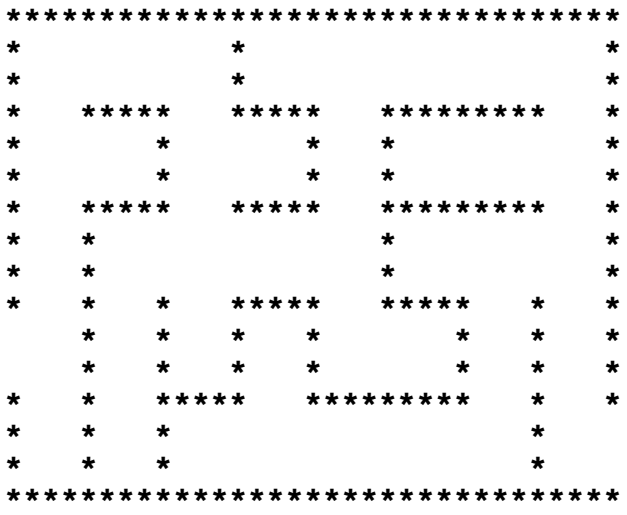
```
 INPUT: Enter X,Y: -4,-18
        Enter X,Y: -15,-18
        Enter X,Y: -15,-2
        Enter X,Y: -4,-2

        Enter A,B: -9,-10
        Enter A,B: -12,-10
        Enter A,B: -12,-8
        Enter A,B: -9,-8

OUTPUT: 6
```

3.9 Write a program to divide two big positive numbers, each with at most 30 digits. The first number will be greater than the second. The program must display the quotient of the first number divided by the second number and the remainder as a whole number. Example:

```
 INPUT: Enter first number: 123456789012345678903
        Enter second number: 1234567890

OUTPUT: 100000000010 REMAINDER 3
```

3.10 Write a program to produce random mazes of dimensions 8 paths by 5 paths. The outer perimeter is 33 asterisks by 16 asterisks. Each horizontal wall segment shall consist of 4 asterisks (from the point of connection), and each vertical wall segment consists of 3 asterisks (from the point of connection). There are 7x4=28 walls that must be displayed. There must not be a wall that is not rooted to the perimeter of the maze. Every area in the maze must be accessible (no closed off areas). On both sides of the maze perimeter, one wall must be removed to allow an outlet. Examples:





FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '88

1.1 Write a program to first clear the screen and then print the phrase:

THE BEST COMPUTER CONTEST!

ten times on the first ten consecutive lines on the screen, each beginning in column 1.

1.2 Write a program to determine if a given number is an INTEGER value or only a REAL value. Example:

```
 INPUT: Enter #: 2.0
OUTPUT: INTEGER

 INPUT: Enter #: -1.5
OUTPUT: REAL
```

1.3 Assume that the surface of a floppy diskette has 40 tracks, and there are 8 sectors per track with 512 bytes/sector. Write a program to determine the number of bytes of information that is on the surface of N floppy diskettes, where N is input as a positive integer. Example:

```
 INPUT: Enter N: 4

OUTPUT: 655360
```

1.4 A complete computer consists of a CPU, PRIMARY memory, SECONDARY memory, an INPUT device, and an OUTPUT device. Given four of the computer's components as input, determine which component is missing. Example:

```
 INPUT: Enter component: SECONDARY
        Enter component: OUTPUT
        Enter component: CPU
        Enter component: INPUT

OUTPUT: PRIMARY
```

1.5 Write a program to outline the perimeter of the screen with asterisks (*) and to divide the screen into four approximately congruent rectangles using *'s. Print the numbers 1, 2, 3, and 4 centered in each rectangle as shown below in miniature.

```
OUTPUT: *************
        *     *     *
        *  1  *  2  *
        *     *     *
        *************
        *     *     *
        *  3  *  4  *
        *     *     *
        *************
```

1.6 Many times a long computer term will be abbreviated by forming an acronym: Electronic Numerical Integrator And Calculator becomes ENIAC. Write a program to make acronyms for given sets of words. Each word in a set will be separated by a space. Acronyms are formed from the first letter of each word in a set. Example:

```
 INPUT: Enter words: RANDOM ACCESS MEMORY
OUTPUT: RAM

 INPUT: Enter words: CENTRAL PROCESSING UNIT
OUTPUT: CPU
```

1.7 Computers are often classified into three categories: MICRO computers, MINI computers, and MAINFRAME computers. MICRO computers are small enough to be placed on a small table and are often used during high school computer programming contests. A MINI computer's CPU and main memory are usually placed on the floor in a case and stand 3 to 5 feet high. MAINFRAME computer systems usually occupy a large room and can be accessed by more than one hundred terminals at the same time. Given the name of a computer and the type of computer, print out a list of the computer names in ascending order according to their size. MAINFRAME is larger than a MINI which is larger than a MICRO. Three computers, each of a different category, will be entered (name, then the type of computer). Example:

```
 INPUT: Enter name: FRED
        Enter type: MINI
        Enter name: WYLBUR
        Enter type: MAINFRAME
        Enter name: GEORGE
        Enter type: MICRO

OUTPUT: GEORGE
        FRED
        WYLBUR
```

1.8 A display in a grocery store will consist of cans stacked on top of each other with the bottom row having N cans in it and each row above it having two cans less than the supporting row. Write a program to find out how many cans will be needed in the display if N is input as the number of cans for the bottom row. N will be greater than 2. Example:

```
 INPUT: Enter N: 5
OUTPUT: 9

 INPUT: Enter N: 6
OUTPUT: 12
```

1.9 A queue is a first-in first-out (FIFO) storage. Objects are removed from a queue in the same order as they are added. A waiting line in a supermarket is an example of a queue. Write a program to place numbers on a queue and then retrieve from or add to the queue. The user is to enter the option to ADD to the queue or TAKE from the queue or QUIT. If the user enters ADD then the program accepts a value to place on the queue. If the user enters TAKE then the first value in the queue is removed and displayed. If QUIT is entered then the program ends. Example:

```
 INPUT: Enter command: ADD
        Enter integer: 5
        Enter command: ADD
        Enter integer: 8
        Enter command: TAKE
OUTPUT: 5
 INPUT: Enter command: ADD
        Enter integer: 2
        Enter command: TAKE
OUTPUT: 8
 INPUT: QUIT
OUTPUT: (program terminates)
```

1.10 Write a program to determine the events of history that occurred between two dates input. Use the following DATA statements in your program (you may store the DATA elements in an array). Each DATA statement contains a year of an invention, person(s) responsible, and invention. Output must display the person followed by the word INVENTED and the invention for each event that is between two years that are input. The first date will precede the second date in time. Example:

```
DATA 1642,"BLAISE PASCAL","ADDING MACHINE"
DATA 1801,"JOSEPH JACQUARD","PUNCHCARD AND WEAVING LOOM"
DATA 1830,"CHARLES BABBAGE","DESIGN OF ANALYTIC ENGINE"
DATA 1890,"HERMAN HOLLERITH","PUNCHCARD TABULATING MACHINE"
DATA 1944,"HOWARD AIKEN","MARK I"
DATA 1946,"ECKERT AND MAUCHLY","ENIAC"
DATA 1949,"VON NEUMAN","EDVAC"
```

INPUT: Enter years: **1640, 1820**

OUTPUT: **BLAISE PASCAL INVENTED ADDING MACHINE**
JOSEPH JACQUARD INVENTED PUNCHCARD AND WEAVING LOOM

2.1 Write a program to display a solid diamond of asterisks of height and length of N, where N is an odd number greater than 1 and less than 22. The middle row has N asterisks and each consecutive row differs by 2 asterisks. The first and last rows each have one asterisk. Example:

```
 INPUT:  Enter N: 5

OUTPUT:    *
          ***
         *****
          ***
           *
```

2.2 Sorting is the arranging of elements of a list into order based on the value of a particular field within the elements. There are several algorithms to accomplish the task of sorting including: bubble sort, shell sort, and quick sort. Each sorting algorithm's efficiency depends upon the number of elements to be sorted and the particular order in which they are originally given. The average number of comparisons needed to sort a list of N elements is given as:

```
N * (N-1) / 2                     for BUBBLE SORT
N * ((Log base 2 of N) squared)   for SHELL SORT
N * (Log base 2 of N)             for QUICK SORT
```

Assuming that the efficiency depends only on the number of elements in the list, N, determine the efficiency order of the different sorting algorithms and print them from most efficient to least efficient. N will be greater than 2. Example:

```
 INPUT: Enter N: 128

OUTPUT: QUICK SORT
        SHELL SORT
        BUBBLE SORT
```

2.3 When a group of no more than 200 was asked to divide itself into groups of 2, 1 person was left over. When asked to divide into groups of 3, 5, 7, there were 2, 1, and 2 left over respectively. Write a program to find out how many people were in this group and display this number.

2.4 Random numbers between 0 and 9999 can be generated according to the following method. Take a 4-digit number as the seed, square it, and pad 0's at the end (if needed) to make the product an 8-digit number. Use the four digits in the middle of the number as the random number and the next seed. Write a program to produce the first 5 random numbers (without leading 0's, if the number is less than four digits) for a given seed of four digits. Example:

```
 INPUT: Enter seed: 3571

OUTPUT: 7520      (because 3571 * 3571 = 12 7520 41 )
        5504      (because 7520 * 7520 = 56 5504 00 )
        2940      (because 5504 * 5504 = 30 2940 16 )
        4360      (because 2940 * 2940 = 86 4360 0 pad 0 )
        96        (because 4360 * 4360 = 19 0096 00 )
```

2.5 Data is often transmitted in 8 bit parcels with the first seven bits being the actual data and the eighth bit being the "parity bit." The parity bit is used to detect errors in transmission. Each bit is either a 0 or a 1. Two techniques for detecting errors are called "odd parity" and "even parity." The transmitter uses the eighth bit to make the eight bit parcel contain an odd number of "1" bits (odd parity) or an even number of "1" bits (even parity). If the transmitter uses odd parity but an even number of "1" bits are received in the transmitted 8 bit parcel, then an ERROR has occurred. If even parity is used, but an odd number of "1" bits are transmitted in the 8 bit parcel then an ERROR has occurred. Write a program to determine if a string of data (of at most 8 characters) contains an ERROR for the indicated parity which is input as ODD or EVEN. The program must also check to see that the string is 8 characters long and contains only "1"s and "0"s. Display the message "ERROR" or "CORRECT" for each input. Example:

```
 INPUT: Enter bits: 00011101
        Enter parity: EVEN
OUTPUT: CORRECT

 INPUT: Enter bits: 11111110
        Enter parity: ODD
OUTPUT: CORRECT

 INPUT: Enter bits: 10101011
        Enter parity: EVEN
OUTPUT: ERROR

 INPUT: Enter bits: 111X1111
        Enter parity: ODD
OUTPUT: ERROR

 INPUT: Enter bits: 110
        Enter parity: EVEN
OUTPUT: ERROR
```

2.6 Write a program that calculates the area of a polygon, using the following formula:

Area = 1/2 * (The absolute value of the following sum):

$$\begin{vmatrix} X1 & Y1 \\ X2 & Y2 \end{vmatrix} + \begin{vmatrix} X2 & Y2 \\ X3 & Y3 \end{vmatrix} + \ldots + \begin{vmatrix} Xn\text{-}1 & Yn\text{-}1 \\ Xn & Yn \end{vmatrix} + \begin{vmatrix} Xn & Yn \\ X1 & Y1 \end{vmatrix}$$

where X and Y are coordinates of the vertices in the X-Y plane, and n is the number of vertices. In words, the AREA is equal to one-half the absolute value of the sum of the corresponding determinants of the coordinates of successive vertices. A determinant of the form: $\begin{vmatrix} X & Y \\ A & B \end{vmatrix}$ is defined as X*B - A*Y, where X,Y and A,B are two successive vertices. The number of vertices, n, will be entered followed by n successive integer coordinates in the X-Y plane. The area is to be displayed in the form ##.#. Example:

INPUT: Enter n: **4**
Enter vertex: **1,0**
Enter vertex: **2,0**
Enter vertex: **2,2**
Enter vertex: **0,4**

OUTPUT: **AREA = 4.0**

2.7 Write a program to accept as input a valid MONTH, DAY, and YEAR between the years 1700 and 1998 inclusive and will display the date of the day before the given date and the date after the given date. A leap year occurs in every year that is divisible by 4 except in those years also divisible by 100. MONTH, DAY, and YEAR will be input as positive integers. The output must be displayed in the form MM-DD-YYYY, with MM and DD displayed without leading zeroes. Example:

INPUT: Enter month, day, year: **3, 1, 1800**

OUTPUT: **2-28-1800**
3-2-1800

2.8 At the University of South Florida, a student may be dismissed at the end of a semester for a low grade point average (GPA). If the student's Cumulative GPA (CGPA) is ever between 0 and 0.999 then he is immediately dismissed. If the student's cumulative GPA is between 1.0 and 1.999 for two consecutive semesters then the student is dismissed. The GPA is determined by assigning points to grades: A-4, B-3, C-2, D-1, F-0. A grade is assigned to each class a student takes. Each class gives from 1 to 5 credit hours. For a given semester the total number of points earned is obtained from summing the points earned for each class, which is obtained by multiplying the grade number by the number of credit hours for that class. The GPA is obtained by dividing the total points earned by the total number of credit hours taken in that semester. The CUMULATIVE GPA is similarly obtained from dividing the total points earned for all semesters by the total number of credit hours taken in all semesters.

Write a program to accept at most 8 semesters of 4 grades with the number of credit hours for each class grade. The program then calculates and displays the student's semester GPA and cumulative GPA in the form #.###, rounded to the nearest 0.001. If the student is dismissed due to a low GPA, then display the message "STUDENT IS DISMISSED" and then end the program. The example INPUT/OUTPUT below is illustrated by calculations within parenthesis that are not a part of the INPUT/OUTPUT.

Example:

```
 INPUT: Enter grade, credits: A, 3        (points = 4x3 = 12)
        Enter grade, credits: B, 3        (points = 3x3 =  9)
        Enter grade, credits: C, 2        (points = 2x2 =  4)
        Enter grade, credits: F, 4        (points = 0x4 =  0)
                                          (total =        25)

OUTPUT:  GPA= 2.083          (because 25 / (3+3+2+4) = 2.0833)
        CGPA= 2.083

 INPUT: Enter grade, credits: F, 5
        Enter grade, credits: D, 2         (points = 1x2 = 2)
        Enter grade, credits: F, 4
        Enter grade, credits: F, 5

OUTPUT:  GPA= 0.125
        CGPA= 0.964           (because (25+2) / (12+5+2+4+5))
        STUDENT IS DISMISSED
```

2.9 Given the elements and their corresponding potentials in DATA statements as shown below, write a program to display the names of all pairs of elements which could be used to produce a battery with voltage equal to the desired VOLTAGE plus or minus the TOLERANCE. The desired battery VOLTAGE and a TOLERANCE will be entered as non-negative real numbers. A battery can be produced if the "difference" in half-reaction oxidation potentials of two elements is less than or equal to the TOLERANCE. If there are no pairs that can form a battery, print the message: "NO BATTERY CAN BE FORMED". The acceptable pairs must be printed with the first element starting in column1, the second element in column 12, and the difference in Potential in column 23 in the form #.##. The order of elements is not important. If there are more than eight acceptable pairs then the first eight should be printed and the user is then prompted with the message, "PRESS ANY KEY FOR MORE". The user then presses a key to see the next eight (or less). If there are still more, then the same process should be repeated until all the pairs have been displayed.

Table of half-reaction oxidation potentials:

```
DATA  ELEMENT  POTENTIAL
---- --------- ---------
DATA "LITHIUM",+3.05
DATA "SODIUM",+2.71
DATA "ZINC",+0.76
DATA "IRON",+0.44
DATA "TIN",+0.14
DATA "IODINE",-0.54
DATA "SILVER",-0.80
DATA "MERCURY",-0.85
DATA "BROMINE",-1.09
DATA "CHLORINE",-1.36
```

Example:

```
 INPUT: Enter Desired Voltage, Tolerance: 5, 1.5

OUTPUT: (in any order)
        LITHIUM    IODINE     3.59
        LITHIUM    SILVER     3.85
        LITHIUM    MERCURY    3.90
        LITHIUM    BROMINE    4.14
        LITHIUM    CHLORINE   4.41
        SODIUM     SILVER     3.51
        SODIUM     MERCURY    3.56
        SODIUM     BROMINE    3.80

        PRESS ANY KEY FOR MORE
 INPUT: (Press any key)
OUTPUT: (continued in any order)
        SODIUM     CHLORINE   4.07
```

2.10 Three local cross country teams compete in a double dual race. Each team consists of seven runners, but only the first five finishers of a team contribute to that team's score. As the runners cross the finish line, gasping for breath, the judge writes the INITIAL of the runner's team name and the NUMBER indicating the runner's finishing position, e.g. 1 for 1st, 2 for 2nd and so on. To find the score for teams A and B and to decide which of the two wins, the scorekeeper temporarily eliminates all of C's positions, and then repositions the runners from A and B into places 1 through 14. The team's score consists of the sum of the places of their first five runners. The lower team score wins. If there is a tie then the team whose sixth runner crossed the finish line first is the winner.

Write a program that computes the score for each pair of three teams and determines the winner of each pair (pairs may be displayed in any order). The program must allow the user to assign all 21 runners' team INITIAL to finishing places. Team initials can be any letter in the alphabet. Example:

```
INPUT: Place 1: A
       Place 2: B
       Place 3: A
       Place 4: B
       Place 5: A
       Place 6: B
       Place 7: C
       Place 8: C
       Place 9: C
       Place 10: C
       Place 11: B
       Place 12: A
       Place 13: C
       Place 14: B
       Place 15: C
       Place 16: B
       Place 17: A
       Place 18: A
       Place 19: C
       Place 20: B
       Place 21: A

OUTPUT: TEAM A: 28 POINTS
        TEAM B: 28 POINTS
        TEAM B WINS!

        TEAM A: 25 POINTS
        TEAM C: 31 POINTS
        TEAM A WINS!

        TEAM B: 24 POINTS
        TEAM C: 31 POINTS
        TEAM B WINS!
```

Example of repositioning teams A and B:

```
 1: A
 2: B
 3: A
 4: B
 5: A
 6: B
 7: B
 8: A
 9: B
10: B
11: A
12: A
13: B
14: A
```

3.1 Write a program to put a set of N real numbers in numerical order, smallest to largest. However, the magnitude of the digits, from smallest to largest, is defined to be 0,8,1,2,5,4,3,9,7,6. Therefore, 810 is less than 172, and -1.5 is less than -8.5. Example:

```
 INPUT: Enter N: 5
        Enter #: 61.2
        Enter #: 801
        Enter #: -5.98
        Enter #: -4.5
        Enter #: 99.99

OUTPUT: -4.5
        -5.98
        99.99
        61.2
        801
```

3.2 A bank can make change for a given amount of money in many different ways. For example, there are 4 ways to make change for 10 cents: 10 pennies, 5 pennies and 1 nickel, 2 nickels, or 1 dime. Write a program to display the total number of ways that change can be made with an input AMOUNT of money using pennies, nickels, dimes, and quarters. AMOUNT will be input in dollars and will be no greater than 2.00. Example:

```
 INPUT: Enter AMOUNT: 0.16

OUTPUT: 6
```

3.3 Write a program to determine if a given point and/or a given cube in space lie/lies "inside" another given cube (CUBE2). The term "inside" includes points or faces shared. The two coordinates of a diagonal of each of the cubes will be input as real numbers in the X-Y-Z coordinate system. Each cube will have its edges parallel to either the X, Y, or Z plane. Output must contain two sentences of the form:

POINT / 1ST CUBE LIES / DOES NOT LIE INSIDE 2ND CUBE

Example:

```
 INPUT: Enter point: 3,4,5
        Enter cube1 diagonal point1:  0,0,0
        Enter cube1 diagonal point2:  3,3,3
        Enter cube2 diagonal point1: -1,0,0
        Enter cube2 diagonal point2:  4,4,4

OUTPUT: POINT DOES NOT LIE INSIDE 2ND CUBE
        1ST CUBE LIES INSIDE 2ND CUBE
```

3.4 Given a set of N letters (where N is between 2 and 5, inclusive), display all the DISTINGUISHABLE permutations of the letters, in alphabetical order. Also display the total number of distinguishable permutations. Note that some of the letters entered could be the same. Example:

```
 INPUT: Enter letters: CABA

OUTPUT: AABC
        AACB
        ABAC
        ABCA
        ACAB
        ACBA
        BAAC
        BACA
        BCAA
        CAAB
        CABA
        CBAA
        TOTAL= 12
```

3.5 Write a program to print a snake (a trail of 25 asterisks '*') centered on the screen. Upon hitting appropriate keys (I--up, J--left, K--right, and M--down), the snake's head moves in the appropriate direction while the rest of the snake slithers along the same right angle paths. The snake is to move CONTINUOUSLY in the designated direction UNTIL a new directional key is hit. The snake will be 25 asterisks long throughout the entire run; Do not leave a sketched path. The snake cannot go backwards, e.g. if it is going to the right, then its next direction cannot be to the left. The snake continues moving until it runs into itself or it runs off the screen or a non-directional key is pressed.

3.6 Write a program to solve a pair of linear equations entered as strings without spaces. The equation will be in one of two forms: AX+BY=C or AX+BY+C=0 where A, B, and C are integers. Assume that the constant value "1" for A and B will be omitted. (e.g. "X" or "Y" instead of "1X" or "1Y"). For "-1", only a "-" will be used. All unnecessary "+" symbols will be omitted. If a unique solution exists then display the solution in the following format:

XSOLUTION= #.# YSOLUTION= #.#

If no solution exists then print NO UNIQUE SOLUTION EXISTS.
Example:

INPUT: Enter equation 1: **X+Y=0**
Enter equation 2: **2X-3Y-4=0**

OUTPUT: **XSOLUTION= 0.8 YSOLUTION= -0.8**

3.7 Write a program to find all semi-perfect numbers less than 35. A number is said to be semi-perfect if there exists a subset of proper divisors of that number that sum to itself. For example, 12 is semi perfect because 1+2+3+6=12 or 2+4+6=12. Next to each semi-perfect number must be the example(s) of how it is semi-perfect. The example(s) are to be ascending order by the factors. If more than one example exists for a semi-perfect number, then display the example with the fewest addends first; if two have the same number of addends, then display the one containing the smallest addend first. Output must be of the following format:

OUTPUT:

SEMI #	**EXAMPLE(S)**
6	**1 + 2 + 3**
12	**2 + 4 + 6**
12	**1 + 2 + 3 + 6**
:	:
:	:

3.8 Write a program to keep score for a bowler. Input will be 10 frames of numbers. Output will be the scoring of each frame, as shown below in the example. The standard method of scoring will be used in this program.

In each frame the bowler has at most two chances to roll down all 10 pins. If the bowler does not knock down all the pins in that frame, then his score is added to the number of pins that he previously knocked down. (Note: the bowler's score starts out as 0).

For the first 9 frames, if the bowler knocks all 10 pins down on the 2nd roll (indicated by a /) then his score for that frame is his previous score + 10 + the number of pins that he knocks down on his next roll in the next frame. If he rolls all the pins down on his 1st ball of a frame (indicated by an X) then he gets 10 + the total number of pins that he knocks down on the next 2 rolls. A new frame is started after 2 balls are rolled or all ten pins are knocked down on the 1st ball.

In the 10th frame, the bowler is allowed at most three rolls, depending upon his first and second rolls. If he gets all the pins down on the 1st roll then his score will be his previous score plus 10 plus the number of pins he knocks down on his next 2 rolls. If he knocks all 10 pins down after 2 rolls, his score will be his previous score + 10 + the number of pins he knocks down on his next roll. Input will be 10 frames, each separated by comma or a space. For the output, each frame input is right justified and the score is left justified in each frame. Example:

INPUT: Enter frames: **7/,X,62,X,X,8/,63,X,X,X9-**

Note: 62 indicates that 6 pins were knocked down on 1st roll
and 2 pins were knocked down on 2nd roll;
/ indicates all remaining pins knocked down on 2nd roll;
X indicates 10 pins knocked down on the 1st roll;
- indicates 0 pins knocked down.

```
OUTPUT: -1- -2- -3- -4- -5- -6- -7- -8- -9- -10-
        ---!---!---!---!---!---!---!---!---!---!
         7/!  X! 62!  X!  X! 8/! 63!  X!  X!X9-!
        20 !38 !46 !74 !94 !110!119!149!178!197!
        ----------------------------------------
```

3.9 Write a program to convert a real number fraction in base M to a real number fraction in base N, where M and N are input as integers between 2 and 16 inclusive. The base M number input will have one whole number on the left of the radix point, ".", and at most seven whole numbers on the right of the radix point. The judging criterion guarantees that the output base N number will always have one whole number on the left of the radix point. Furthermore, there will be at most seven whole numbers, NDigits, on the right of the radix point as determined by the following formula:

```
(1/N) ** NDigits  less than or equal to  (1/M) ** MDigits

  where N, M are input as bases;
  ** represents "raised to the power of"
  MDigits is the number of digits to the right of the input
  fraction;
  NDigits is the smallest integer that satisfies the equation.
```

For example:

```
  INPUT: Enter M, N, #: 2, 5, 1.11011

  (1/5)**NDigits  less than or equal to  (1/2)**5
  (1/5)**NDigits  less than or equal to  1/32
          NDigits = 3 because
     1/5 ** 2 = 1/25   and   1/5 ** 3 = 1/125
  Therefore 3 digits will be to the right of the output
  number, with the 3rd digit ROUNDED TO THE NEAREST
  NUMBER based upon the 4th digit.

  .11011 base 2 = ( 1/2 + 1/4 + 1/16 + 1/32) base 10
                = ( .84375 ) base 10

  .84375 base 10 = ( .4102) Base 5
                 = ( 4/5 + 1/25 + 0/125 + 0/625) base 10

 OUTPUT: 1.410    (Note: 02 ROUNDED DOWN TO 0 because
                  2 is less than N/2=2.5)


  INPUT: Enter M, N, #: 16, 10, 8.FC2

  (1/10) ** NDigits  less than or equal to (1/16)**3
  Therefore, NDigits is 4.

  .FC2 base 16 = 15/16 + 12/256 + 2/4096 = .98486 base 10

 OUTPUT: 8.9849   (Note: 86 ROUNDED UP TO 9 because
                  6 is greater than or equal to N/2)
```

3.10 Write a program to display the composition of two polynomials, given as input. A polynomial is a mathematical expression consisting of constants multiplied by variables raised to a non-negative integral power. The following are examples of polynomials, where ** stands for "to the power of:"

```
p(x)=5x**2 + 4x - 1

q(x)=x**2 - 6
```

The composition of p of q, p(q(x)), is

```
5*(x**2 - 6)**2 + 4*(x**2 - 6) - 1
5*(x**4 - 12x**2 + 36) + 4*(x**2 - 6) - 1
5x**4 - 60x**2 + 180 + 4x**2 - 24 - 1
5x**4 - 56x**2 + 155
```

Two polynomials will be input. First the ORDER (highest power) of the polynomial is entered as an integer from 0 to 5. The coefficient of each term is entered starting with the highest power, then next highest power, down to the 0th power. The second polynomial is entered the same way.

The output will consist of the composite function of the first of the second polynomial p(q(x)), and then the composite function of the second of the first polynomial q(p(x)). Starting from the highest ORDER of the composite function, the output will be of the following format:

```
AX**N + BX**(N-1) + ... + CX**1 + DX**0.
```

For example:

```
 INPUT: Enter the ORDER of p(x): 2
        Enter coefficient for x**2: 5
        Enter coefficient for x**1: 4
        Enter coefficient for x**0: -1

        Enter the ORDER of q(x): 2
        Enter coefficient for x**2: 1
        Enter coefficient for x**1: 0
        Enter coefficient for x**0: -6

OUTPUT: P(Q(X))= 5X**4 + 0X**3 + -56X**2 + 0X**1 + 155X**0
        Q(P(X))= 25X**4 + 40X**3 + 6X**2 + -8X**1 + -5X**0

 INPUT: Enter ORDER of p(x): 0
        Enter coefficient for x**0: 9

        Enter the ORDER of q(x): 1
        Enter the coefficient of x**1: -4
        Enter the coefficient of x**0: 3

OUTPUT: P(Q(X))= 9X**0
        Q(P(X))= -33X**0
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '89

1.1 Write a program to print the phrase 1989 COMPUTER CONTEST on each line of the screen such that each successive phrase is indented one extra space. Example:

```
OUTPUT:  1989 COMPUTER CONTEST
          1989 COMPUTER CONTEST
           1989 COMPUTER CONTEST
            :
             :
              :
```

1.2 A database is a collection of data, or information representing abstract entities. Databases require much storage space. Most businesses measure their databases in terms of gigabytes of data. A gigabyte is equivalent to 1024 or 2^10 megabytes (a megabyte is approximately a million bytes). Write a program to represent a gigabyte value (a gigabyte is approximately a billion bytes) in its equivalent number of megabytes. Input will consist of a positive integer less than 30. Example:

```
 INPUT: Enter number of gigabytes: 14
OUTPUT: 14336 MEGABYTES
```

1.3 Write a program to display a word in a backward-L format. The given word is to be displayed vertically and horizontally so that they share the same last letter. Example:

```
 INPUT: Enter word: EXAMPLE

OUTPUT:       E
              X
              A
              M
              P
              L
         EXAMPLE
```

1.4 Write a program to produce the following pattern for an input integer, N, with N between 1 and 9 inclusive:

```
          1                Example:    INPUT: Enter N: 4
        2 2                           OUTPUT:     1
       3   3                                      2 2
      .     .                                    3   3
     .       .                                  4     4
    N         N
```

1.5 Anno Domini is Latin for "in the year of our Lord" and is abbreviated as A.D.. In 525 A.D., Pope John I asked the monk Dionysius Exiguus to begin a Christian system of dating events, starting with the year Dionysius believed Christ was born. The years before the birth of Christ are termed B.C., while the years after the birth of Christ are termed A.D. The following is the order of years: ... 2 B.C., 1 B.C., 1 A.D., 2 A.D., ... with Christ's birth being in the year 1 A.D. Today, we know that the monk was in error. Even though we continue to use his dating system, biblical scholars currently believe that Christ was born four years earlier than what the monk thought. Write a program that corrects modern dates to account for this change. Examples:

```
 INPUT: Enter date: 4
        Enter A.D. or B.C.: B.C.
OUTPUT: 1 A.D.

 INPUT: Enter date: 1
        Enter A.D. or B.C.: A.D.
OUTPUT: 5 A.D.

 INPUT: Enter date: 5
        Enter A.D. or B.C: B.C.
OUTPUT: 1 B.C.
```

1.6 Many computer systems require a user to enter a password to ensure that the appropriate person is accessing his/her files of information. Write a program to prompt a user for a password with the words "ENTER PASSWORD: ". For this program the user's password is ITSME. The user has up to 3 chances to enter the correct password. If it is correct then display the message "YOU HAVE ACCESS". For the first two tries, if an incorrect password is entered, display "INVALID PASSWORD:" and prompt for another password. After 3 incorrect entries, display "YOU ARE TRESPASSING" and exit. Example:

```
OUTPUT/INPUT:  ENTER PASSWORD: TRY1
OUTPUT/INPUT:  INVALID PASSWORD: TRY2
OUTPUT/INPUT:  INVALID PASSWORD: TRY3
OUTPUT:        YOU ARE TRESPASSING

OUTPUT/INPUT:  ENTER PASSWORD: TRYAGAIN
OUTPUT/INPUT:  INVALID PASSWORD: ITSME
OUTPUT:        YOU HAVE ACCESS
```

1.7 Write a program to determine the "best" Data Base Management System (DBMS) to use. A DBMS is a set of programs that access a collection of interrelated data, called a database. A DBMS is "good" if it provides an environment that is both convenient and efficient to use in retrieving data from the database and in storing data into the database.

First, N will be input as the number of DBMS to consider. Next, N names pertaining to the DBMS will be entered, each followed by it's respective convenience rank and efficiency rank (both between 1 and 10 inclusive). The "best" DBMS is determined by the highest total rank for convenience and efficiency. Display the name of the best DBMS that "IS BEST". In the example below, Amy is best because (3+9) is larger than (10+1) which is larger than (3+4). Example:

```
 INPUT:  Enter N: 3
         Enter DBMS name: DOUG
         Enter convenience, efficiency: 3, 4
         Enter DBMS name: AMY
         Enter convenience, efficiency: 3, 9
         Enter DBMS name: CRAIG
         Enter convenience, efficiency: 10, 1
OUTPUT:  AMY IS BEST
```

1.8 Write a program to display the elements of a list of integers, without repetition, in the order of their appearance in the list, separated by one space. One number at a time will be input. Termination of the list will be designated by the input of -999. Example:

```
 INPUT: Enter #: 2
        Enter #: 3
        Enter #: -1
        Enter #: 3
        Enter #: 2
        Enter #: -5
        Enter #: -999

OUTPUT: 2 3 -1 -5
```

1.9 Often statisticians compute such large probabilities that they are difficult for the average person to comprehend. To help illustrate such a number, a real life model is used. Write a program to illustrate the probability of "1 out of N", where N is a large real number given in scientific "E" notation. Output will consist of the nearest integer of FEET DEEP of silver dollars that the state of Texas needs to be covered to be equivalent to the number N. Output will be less than 1000.

Assume that Texas has 262,134 square miles of land, while a silver dollar is 1 1/2 inches in diameter and 3/32 inch in thickness. Assume that the silver dollars will be piled in rows and columns. Examples:

```
 INPUT: Enter probability: 1E17
OUTPUT: 2 FEET DEEP

 INPUT: Enter probability: 2.6E18
OUTPUT: 43 FEET DEEP
```

silver dollars are piled like:

OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO
OOOOOOOOOO

1.10 Memory is a large array of bytes, each with its own address. Each program in a computer system deals with particular logical addresses. The memory mapping hardware converts logical addresses into physical addresses. Logical addresses are in the range of 0 to Max. The corresponding Physical addresses are in the range of R+0 to R+Max, where the value R is the lower bound.

Write a program to map a given logical address and segment to the corresponding physical address. Each segment starts at a specific physical address (base) and has a given length. The physical address can be determined using the data in the table below by adding the base (smallest physical address in a specified segment) to the given logical address. If the logical address specified is greater than the length of the segment, display ADDRESSING ERROR. Input will be the segment number followed by the logical address to convert. Output will be an error message (ADDRESSING ERROR) or the physical address. Repeat input until a segment number greater than 4 is entered.

Data:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Example:

```
 INPUT: Enter Seg#, Address: 1, 11
OUTPUT: 2311
 INPUT: Enter Seg#, Address: 3, 600
OUTPUT: ADDRESSING ERROR
 INPUT: Enter Seg#, Address: 2, 95
OUTPUT: 185
 INPUT: Enter Seg#, Address: 5, 0
OUTPUT: (program terminates)
```

2.1 Write a program to display the value of F(x), given x as an input positive integer between 1 and 10 inclusive, and given:

$$F(1) = F(2) = F(3) = 1, \text{ and}$$

$$F(x+1) = \frac{F(x)*F(x-1)+2}{F(x-2)} \quad \text{for x greater than 3}$$

Output must be of the form "F(x)=" F(x), where the first x is substituted by the input value of x, and the second F(x) is the actual value of the function. Examples:

```
 INPUT: Enter x: 4              INPUT: Enter x: 6
OUTPUT: F(4)= 3                OUTPUT: F(6)= 17
```

2.2 Write a program to print out the prime factorization equation for a given positive integer. The prime numbers must be in increasing order and separated by an "X". Examples:

```
 INPUT: Enter #: 90             INPUT: Enter #: 17
OUTPUT: 2 X 3 X 3 X 5          OUTPUT: 17
```

2.3 Write a program to display a word without its vowels: a,e,i,o,u, and exclude y. Example:

```
 INPUT: Enter word: CONTEST
OUTPUT: CNTST


 INPUT: Enter word: ANSWER
OUTPUT: NSWR
```

2.4 In order to write a program, a programmer must choose appropriate names for variables, constants, procedures, etc. Each identifier should have a name that correctly identifies its purpose and function in the program. However, if a name is too long, then it is burdensome to write and read in a program and may occupy more space on a line than is necessary. Good short names that properly describe an object's function are better than long names.

For the sake of brevity, write a program to produce the shortest possible names for a set of 6 identifiers in a program so that each one is distinguishable using the following method. If two or more identifiers start with the same character(s) compare each identifier character by character until they are distinguishable. Truncate the remainder of the identifier after the distinguishable character (see MINIMUM, MAXIMUM, and MAXNUM in example). Assume that the language can distinguish identifiers with any amount of letters, if they differ. Example:

```
INPUT: Enter name: AVERAGE          OUTPUT: A
       Enter name: MINIMUM                  MI
       Enter name: MAXIMUM                  MAX
       Enter name: MAXNUM                   MAXN
       Enter name: POSITION                 POS
       Enter name: POINTER                  POI
```

2.5 Write a program to display the number of distinguishable permutations of letters of a given word. The mathematical formula for calculating such a number is given as the factorial, (!), of the number of letters in the word divided by the product of factorials of the number of times a letter appears. Examples:

```
 INPUT: Enter word: WITTICISM
OUTPUT: 30240
```

(there are 9 letters, 3 I's, and 2 T's, 9! / (3! x 2!) = 30240)

```
 INPUT: Enter word: ELEMENT
OUTPUT: 840
```

(there are 7 letters, 3 E's, 7! / 3! = 840)

NOTE: A factorial is the product of all the integers from 1 to the number. (e.g. 5!=5x4x3x2x1 = 120) Since 1! = 1, letters that appear only once are not relevant.

2.6 Write a program to simulate a word processor that underlines text between two asterisks. Underlining mode begins at the first asterisk and ends at the next asterisk. Many different segments of a line may be underlined. The line of text will contain no more than 40 characters. The program is to accept a line of text, clear the screen, display the line, skip a line, display the line without it's embedded asterisks, then underline (with hyphens on the next line) the words between the first and second asterisks, the third and fourth, and so on. Example:

```
 INPUT: I *REALLY* THINK THAT *WE WILL WIN*

OUTPUT: (Screen is cleared)
        I *REALLY* THINK THAT *WE WILL WIN*

        I REALLY THINK THAT WE WILL WIN
          ------           -----------
```

2.7 Write a program to compute an integer expression involving two positive integers separated by one of the following: +, -, *, or /. Each integer will contain no more than 4 digits and the result is guaranteed to be an integer between -30,000 and 30,000. Examples:

```
 INPUT: 80/5        INPUT: 543*21        INPUT: 999-5556
OUTPUT: 16         OUTPUT: 11403        OUTPUT: -4557
```

2.8 Game theory is a mathematical theory formally dealing with competitive situations. Emphasis is placed on the decision-making processes of the adversaries. In a two-person game, a player may use a two dimensional matrix of numbers to represent a payoff table. Each number represents an amount of win or loss. Each player tries to minimize his maximum losses. If there exists an element in the table that is both the minimum in its row and the maximum in its column, then it is called a saddle point. When a saddle point exists, neither player has an advantage over his opponent.

Write a program to determine the saddle point element (each set of data will have a saddle point) and its row position and column position. The program is to first accept the number of rows and columns in the table. Then, the program accepts the elements of the table starting with each column element in row 1, then each column element of row 2, etc. The output must be of the form, SADDLE POINT = # AT ROW # COL #. Example:

```
INPUT: Enter # Rows, # Cols: 3, 3
       Enter Row1 Col1: -3
       Enter Row1 Col2: -2
       Enter Row1 Col3: 6
       Enter Row2 Col1: 2
       Enter Row2 Col2: 0
       Enter Row2 Col3: 2
       Enter Row3 Col1: 5
       Enter Row3 Col2: -2
       Enter Row3 Col3: -4

OUTPUT: SADDLE POINT = 0 AT ROW 2 COL 2
```

2.9 Write a program to sort a set of dates entered. First, accept the number of dates to sort. Next, accept each date by first accepting the entire name of the month, then the day, then the year as integers. Display the dates in order in the form: MONTH DAY YEAR. Example:

```
INPUT: Enter # of dates: 3
       Enter month: JANUARY
       Enter day:   1
       Enter year:  1978

       Enter month: FEBRUARY
       Enter day:   20
       Enter year:  1977

       Enter month: JANUARY
       Enter day:   27
       Enter year:  1978

OUTPUT: FEBRUARY 20 1977
        JANUARY 1 1978
        JANUARY 27 1978
```

2.10 Because Ms. Heindel is such a nice music teacher, she is allowing her students to retake quiz 4, since her class did rather poorly. Write a program to display the table of names and grades (with column headings) as shown below, then accept the 5 new grades for the students for quiz 4, in their order of listing. Then clear the screen and display the final report with averages for each person and for each quiz and for the overall class. Averages must be accurate to 2 decimal places and be aligned in the proper column. The headings in the final report must be centered, as shown below. With the exception of spacing, the content must look as follows:

RUN PROGRAM:

OUTPUT:

NAME	Q1	Q2	Q3	Q4
D. WOOLY	100	92	90	90
M. SMITH	55	75	70	65
C. BROWN	94	70	62	70
R. GREEN	90	74	80	85
T. STONE	85	98	100	70

INPUT: Enter 5 grades for quiz 4: **98, 70, 75, 90, 80**

OUTPUT: (Screen is cleared)

MS. HEINDEL'S MUSIC CLASS
FINAL GRADES
SPRING 1989

NAME	Q1	Q2	Q3	Q4	AVERAGE
D. WOOLY	100	92	90	98	95.00
M. SMITH	55	75	70	70	67.50
C. BROWN	94	70	62	75	75.25
R. GREEN	90	74	80	90	83.50
T. STONE	85	98	100	80	90.75
AVERAGE:	84.80	81.80	80.40	82.60	

CLASS AVERAGE: 82.40

3.1 Write a program to simulate a mini spell checker. Given a word, determine if it is CORRECT or MISSPELLED. A word is misspelled if it violates any of the following spelling rules:

- 'E' does not appear before the suffixes 'ING', 'IBLE', 'ABLE'
- 'I' before 'E' except after 'C'
- A letter may appear no more than twice consecutively

Examples:

INPUT: Enter word: **APPPLE**	OUTPUT: **MISSPELLED**
INPUT: Enter word: **PIE**	OUTPUT: **CORRECT**
INPUT: Enter word: **EATING**	OUTPUT: **CORRECT**
INPUT: Enter word: **NOTEABLE**	OUTPUT: **MISSPELLED**
INPUT: Enter word: **RECIEVE**	OUTPUT: **MISSPELLED**

3.2 Write a program to calculate the positive amount of (V)olume for a given (P)ressure according to the following thermo-dynamics equation:

P*V*V*V*14.14 - P*V*9062.599 - 23511.9*V*V + 988686.1*V =400943.0

The program will first simulate the values of V for the following values of P: 0.05, 0.70, 10.00, 70.00. Next, a positive value of P (less than 100) is entered and the value of V is computed and displayed. Values of V must be rounded to the nearest ten-thousandth (0.0001). Display both the value of P and V, as shown below, where ? represents the computed value for V. Example:

RUN PROGRAM:

OUTPUT: **P = 0.05 V = 0.4097**
P = 0.70 V = ?
P = 10.00 V = ?
P = 70.00 V = 1.2263
INPUT: Enter value for P: **30.00**
OUTPUT: **P = 30.00 V = 0.5699**

3.3 Write a program to magnify an input positive integer. Each digit must be displayed in block format (made of *) with dimensions determined by its magnification (1, 2, or 3). At most 4 digits will be input for magnification of 1; At most 3 digits for magnification of 2, and at most 2 digits for magnification of 3. The dimensions of a block number are as follows:

Magnification	Dimensions	Space Between Digits
1	5 rows by 4 columns	(2 space separation)
2	9 rows by 8 columns	(4 space separation)
3	13 rows by 12 columns	(6 space separation)

Examples:

```
 INPUT: Enter number: 3124
        Enter magnification: 1

OUTPUT: ****     *  ****  *  *        (note: digit 1 is in
           *     *     *  *  *         right most column of
        ****     *  ****  ****         block format)
           *     *  *        *
        ****     *  ****     *
```

```
 INPUT: Enter number: 567
        Enter magnification: 2

OUTPUT: ********    ********    ********
        *           *                  *
        *           *                  *
        *           *                  *
        ********    ********           *
               *    *      *           *
               *    *      *           *
               *    *      *           *
        ********    ********           *
```

```
 INPUT: Enter number: 89
        Enter magnification: 3

OUTPUT: ************      ************
        *          *      *          *
        *          *      *          *
        *          *      *          *
        *          *      *          *
        *          *      *          *
        ************      ************
        *          *                 *
        *          *                 *
        *          *                 *
        *          *                 *
        *          *                 *
        ************                 *
```

3.4 Write a program to produce a calendar for an input month of a year. The month will be any number between 1 and 12 inclusive, and the year will be between 1901 and 1999 inclusive. Every year (in this century) divisible by 4 is a leap year. January 1, 1901 was a Tuesday. Each heading of the month and year will be centered above the calendar. The rest of the calendar should appear exactly as shown below. Sunday's column has a 2 space margin; all the other days have a 4 space margin in which the numbers are right justified. Examples:

```
 INPUT: Enter month, year: 1, 1901
OUTPUT:          JANUARY 1901
           S   M   T   W   T   F   S
          --------------------------
                      1   2   3   4   5
           6   7   8   9  10  11  12
          13  14  15  16  17  18  19
          20  21  22  23  24  25  26
          27  28  29  30  31

 INPUT: Enter month, year: 2, 1988
OUTPUT:         FEBRUARY 1988
           S   M   T   W   T   F   S
          --------------------------
                  1   2   3   4   5   6
           7   8   9  10  11  12  13
          14  15  16  17  18  19  20
          21  22  23  24  25  26  27
          28  29
```

3.5 Write a program to determine all the possible ways to position 5 queens on a 5x5 chess board so that none of them can attack another. A queen can attack another queen if the other queen is in the same row or column or diagonal. For each solution, display the column the queen is in pertaining to the row. Output display must be of the format shown below, with each column number in row 1 non-decreasing for each solution. If 2 solutions have the same column number in row 1, put the solution with the smallest column number in row 2 first.

```
OUTPUT:  ROWS =  1 2 3 4 5
         -----------------
         COLUMNS
                 1 3 5 2 4
                 : : : : :
                 : : : : :
                 5 3 1 4 2
```

The first solution has queens in (row 1, col 1), (row 2, col 3), (row 3, col 5), (row 4, col 2), (row 5, col 4). The last solution has queens in (row 1, col 5), (row 2, col 3), (row 3, col 1), (row 4, col 4), (row 5, col 2).

3.6 Write a program to display the product of two large integers in a given base. Integers will be at most 30 digits in length, and the base will be between 2 and 10 inclusive (both the input integers and output integers will be in the given base). Examples:

```
 INPUT: Enter base: 8
        Enter first integer: -12345670123456701234567
        Enter second integer: 7654321076543210
OUTPUT: -121705336146616716573067044023333510470

 INPUT: Enter base: 10
        Enter first integer:  1234567890
        Enter second integer: 9999999999
OUTPUT: 12345678898765432110
```

3.7 Write a program to determine the most efficient way to represent change. Input will consist of the COST, the given AMOUNT, and the COIN that is unavailable for making change. Pennies, nickels, dimes, and quarters are available for making change, with the exception of the COIN input as missing. The most efficient change is defined as the combination that requires the fewest number of coins.

Output must display the change returned, starting with the smallest denomination, and not including the missing coin. Next, the total change returned is displayed. Change returned will not exceed 99 cents. If only one coin of a denomination is used, then the singular form of the name must be used (as in PENNY). Otherwise, the plural form is used (as in PENNIES). Examples:

```
 INPUT: Enter cost, amount: 14.41, 15.00
        Enter missing coin: NICKEL
OUTPUT: 4 PENNIES
        3 DIMES
        1 QUARTER
        TOTAL CHANGE RETURNED = 59 CENTS

 INPUT: Enter cost, amount: 4.40, 5.00
        Enter missing coin: DIME
OUTPUT: 0 PENNIES
        2 NICKELS
        2 QUARTERS
        TOTAL CHANGE RETURNED = 60 CENTS
```

3.8 Write a program to find the corner coordinates of rectangles consisting of 1's in a two dimensional table of binary numbers. The table consists of 6 rows by 7 columns. Six numbers in base 10 (each less than 128) are entered for each row. These numbers are converted into base 2 and padded with 0's on the left (if necessary) to fit into the 7 columns of that row. The program will display this table. Next, the computer finds all rectangles filled with 1's and displays the coordinates of the upper left corner and the bottom right corner of each rectangle. A rectangle must have dimensions of at least 2. No rectangles will overlap (i.e., in cases where rectangle is contained in another rectangle, give the coordinates of the larger rectangle). Display each set of coordinates on a separate line with parenthesis and commas, as shown below. Example:

```
INPUT: Enter number: 127          OUTPUT: 1111111
       Enter number: 123                  1111011
       Enter number: 23                   0010111
       Enter number: 99                   1100011
       Enter number: 88                   1011000
       Enter number: 57                   0111001

                                          (1,6)(4,7)
                                          (1,1)(2,4)
                                          (5,3)(6,4)
```

(Note: 3 lines of coordinates may appear in any order)

3.9 Write a program to determine the five word combination that gives the greatest point value according to the following rules. Two sets of five words each (shown below in the first output) are given as data. The words in each set are placed on top of each other in such a way that BINGO is spelled in the first column of the first set of words and the second column of the second set of words. Each word has a numerical value as determined by summing the values of each letter as shown below:

```
 A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P
 9 14  1 16 20  5 10  2 21 17  6 25 12  3 22 18

 Q  R  S  T  U  V  W  X  Y  Z
24  7 13 26 15 11 19  4 23  8
```

The sum of the letters in each word is displayed to the right of the word and a total of all the word sums is placed beneath the five word sums. The set of five words with the greatest total has 3 asterisks placed underneath its total.

The program will start with the 10 given words displaying the numerical sums. Next, the program will accept any number of new five letter words to replace those words in the original two sets. For a new word to replace another word currently in the set, its total must be larger than the total of the word it is replacing (a new word may be used in either or both lists). Also, the entire set of words must still spell BINGO in the first or second columns as it did previously. Pressing the <ENTER> or <RETURN> key will end the input of words and display the new word list and sums (with the larger total sum underlined with three asterisks). The program then accepts more words, or it may quit if the word QUIT is entered. The second set of five words (with BINGO down the second column) must be placed to the right of the first set of words, as shown below. Example:

```
OUTPUT: BIBLE  94   OBESE  89
        IDYLL 110   TITHE  95
        NOISE  79   INLET  95
        GULLY  98   IGLOO 100
        OBESE  89   TOWER  94
              470         473
                          ***
 INPUT: Enter word: NOTED
        Enter word: BOOST
        Enter word: OLIVE
        Enter word: ONION
        Enter word: (Enter key pressed)
OUTPUT: BOOST  97   OBESE  89
        IDYLL 110   TITHE  95
        NOTED  87   INLET  95
        GULLY  98   IGLOO 100
        OLIVE  99   BOOST  97
              491         476
              ***
  INPUT: Enter word: QUIT
 OUTPUT: (program terminates)
```

3.10 Write a program to determine the number of distinguishable ways to place and orient a cube with a solid color on each of its six sides. The program will ask for the one letter color symbol for each of the sides in the following order: TOP, FRONT, BOTTOM, BACK, RIGHT, LEFT. Output will be a statement declaring the NUMBER OF DISTINGUISHABLE CUBES--different ways to position and orient the cube. Examples:

```
 INPUT: Enter TOP side:    Y
        Enter FRONT side:  Y
        Enter BOTTOM side: Y
        Enter BACK side:   Y
        Enter RIGHT side:  Y
        Enter LEFT side:   Y
OUTPUT: NUMBER OF DISTINGUISHABLE CUBES = 1

 INPUT: Enter TOP side:    G
        Enter FRONT side:  G
        Enter BOTTOM side: G
        Enter BACK side:   G
        Enter RIGHT side:  G
        Enter LEFT side:   Y
OUTPUT: NUMBER OF DISTINGUISHABLE CUBES = 6

 INPUT: Enter TOP side:    B
        Enter FRONT side:  B
        Enter BOTTOM side: O
        Enter BACK side:   Y
        Enter RIGHT side:  G
        Enter LEFT side:   R
OUTPUT: NUMBER OF DISTINGUISHABLE CUBES = 24
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '90

1.1 Write a program to produce the following initials for NCNB National Bank:

```
NN    N  CCCCC  NN    N   BBBB
N N   N  C      N N   N   B   B
N  N  N  C      N  N  N   BBBBB
N   N N  C      N   N N   B   B
N    NN  CCCCC  N    NN   BBBB
```

1.2 NCNB programmers in Tampa use an IBM 3090 model 400 mainframe computer that has an MVS/XA (Multiple Virtual Storage/ Extended Architecture) operating system. The machine is partitioned into two logical processors: SYSTEM 1 is used for testing and all day-time batch jobs; SYSTEM 2 is used to support on-line programs and some night-time batch jobs. Write a program to display the processor name given the system #. Examples:

```
 INPUT: Enter #: 1          INPUT: Enter #: 2
OUTPUT: SYSTEM 1           OUTPUT: SYSTEM 2
```

1.3 With assets over 66 billion dollars in 1989, NCNB is the largest bank in the southeast United States. Let's assume that the Tampa Application Systems Center staffs 66 programmers. If there is a direct correlation between the number of programmers staffed in Tampa and the amount of money the bank has in assets, then display the total amount of assets NCNB will have if N more programmers are hired. Example:

```
 INPUT: Enter N: 4
OUTPUT: 70 BILLION DOLLARS
```

1.4 Write a program to help the post office determine the county in which a person lives, given his/her zip code. Use the following as data:

```
HILLSBOROUGH - 33612  33613  33620  33510
PINELLAS -     33701  34685  34646
PASCO -        33525  34249  34690
```

Example:

```
 INPUT: Enter zip code: 33525
OUTPUT: PASCO
```

1.5 NCNB's dynamic chairman, Hugh McColl, resides at the national headquarters in Charlotte, North Carolina. His secretary would like to issue a statement to all the employees concerning McColl's financial goals. Write a program that accepts two numbers (MMM - amount in billions of dollars, and YYYY - year), and then produces a statement in the following form:

```
HUGH MCCOLL WOULD LIKE NCNB TO GROW
TO MMM BILLION DOLLARS IN ASSETS BY
THE YEAR YYYY
```

Example:

```
 INPUT: Enter MMM: 100
        Enter YYYY: 1995

OUTPUT: HUGH MCCOLL WOULD LIKE NCNB TO GROW
        TO 100 BILLION DOLLARS IN ASSETS BY
        THE YEAR 1995
```

1.6 Vickie manages 7 people in the Trust Division at NCNB. Her associates pay an average of 50,000 coupons monthly. The work load is divided as evenly as possible among her associates so that no one person has more than one extra coupon to pay than another person. Write a program to calculate the MAXIMUM number of coupons that any one of N associates must pay if C coupons come in for payments. Examples:

```
 INPUT: Enter N associates: 7
        Enter C coupons: 49000
OUTPUT: 7000

 INPUT: Enter N associates: 7
        Enter C coupons: 49002
OUTPUT: 7001
```

1.7 NCNB programmers code their programs primarily in COBOL: COmmon Business Oriented Language. Each program is required to have these four divisions (in order): IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE. Write a program to display those divisions coming before and after a division given as input. Multiple divisions on a line are to be displayed in order and with 2 spaces in between. If none occur before/after entered division, then display NONE for that part. Examples:

```
 INPUT: Enter division: DATA
OUTPUT: BEFORE = IDENTIFICATION  ENVIRONMENT
        AFTER = PROCEDURE

 INPUT: Enter division: IDENTIFICATION
OUTPUT: BEFORE = NONE
        AFTER = ENVIRONMENT  DATA  PROCEDURE
```

1.8 NCNB is the largest banking company in the South and the 7th largest in the nation. NCNB has statewide banks in four states: Florida, North Carolina, South Carolina, and Texas. It also has banks in Baltimore, Atlanta, and northern Virginia. For 1990, NCNB will recognize the following number of holidays in each of the states: 10-FL, 8-NC, 7-SC, 10-TX, 11-MD, 10-GA, 10-VA. Write a program to display the states that recognize at least N holidays, where N is input as a number between 5 and 11 inclusive. States must be displayed in the order given above with one space in between each state. Example:

```
 INPUT: Enter N: 10
OUTPUT: FL TX MD GA VA
```

1.9 Anno Domini is Latin for "in the year of our Lord" and is abbreviated as A.D. In 525 A.D., Pope John I asked the monk Dionysius Exiguus to begin a Christian system of dating events, starting with the year Dionysius believed Christ was born. The years before the birth of Christ are termed B.C., while the years after the birth of Christ are termed A.D. The following is the order of years: ... 2 B.C., 1 B.C., 1 A.D., 2 A.D., ... with Christ's birth being in the year 1 A.D. Today, we know that the monk was in error. Even though we continue to use his dating system, biblical scholars currently believe that Christ was born four years earlier than what the monk thought. Write a program that corrects modern dates to account for this change. Examples:

```
 INPUT: Enter date: 4
        Enter A.D. or B.C.: B.C.
OUTPUT: 1 A.D.

 INPUT: Enter date: 1
        Enter A.D. or B.C.: A.D.
OUTPUT: 5 A.D.

 INPUT: Enter date: 5
        Enter A.D. or B.C.: B.C.
OUTPUT: 1 B.C.
```

1.10 Write a program to display a word diamond for a 7 letter word. The following examples will illustrate the format of a word diamond. Examples:

```
 INPUT: Enter word: CONTEST

OUTPUT:      T
            NTE
           ONTES
          CONTEST
           ONTES
            NTE
             T
```

```
 INPUT: Enter word: PROBLEM

OUTPUT:      B
            OBL
           ROBLE
          PROBLEM
           ROBLE
            OBL
             B
```

2.1 Write a program to encode a phrase for the Security department. Each letter in the phrase is to be replaced by the letter that precedes it in the alphabet (B becomes A, C becomes B, ... Z becomes Y), except that the letter A becomes Z. All other characters remain the same. Example:

INPUT: Enter phrase: **THIS PERSON IS A THIEF**

OUTPUT: **SGHR ODQRNM HR Z SGHDE**

2.2 Write a program to determine if a given year is the BEGINNING/END of a DECADE/CENTURY/MILLENNIUM. Year 1 is the first year of our calendar, the year traditionally looked on as the year of Christ's birth. Year 1 began the first decade and the first century and the first millennium. The year 2000 will be the end of a decade and a century and a millennium. Input will be a year between 1 and 2000 inclusive. Output will be all valid possibilities of the form XXX OF YYY, where XXX is either BEGINNING or END, and YYY is either DECADE, CENTURY, or MILLENNIUM. For more than one line of output, the order must be DECADE, CENTURY, MILLENNIUM. Examples:

INPUT: Enter year: **1900** OUTPUT: **END OF DECADE**
END OF CENTURY

INPUT: Enter year: **1991** OUTPUT: **BEGINNING OF DECADE**

2.3 Bob, Doug, Jackie, and Jose bowl in the NCNB bowling league. Given the scores of each of their 3 games, display each person's average and handicap. If a person has an average over 200, then his handicap is 0. Otherwise, the handicap is calculated by subtracting an average from 200 and multiplying the result by 90%. Both the average and handicap must be truncated to a whole number when displayed. Example:

INPUT: Enter scores for Bob: **200, 190, 218**
Enter scores for Doug: **195, 207, 168**
Enter scores for Jackie: **109, 134, 127**
Enter scores for Jose: **130, 140, 144**

OUTPUT: **BOB: AVERAGE = 202 HANDICAP = 0**
DOUG: AVERAGE = 190 HANDICAP = 9
JACKIE: AVERAGE = 123 HANDICAP = 69
JOSE: AVERAGE = 138 HANDICAP = 55

2.4 Today we use a Gregorian Calendar, initially set up by Pope Gregory XIII in 1582. Because the previous Julian calendar was not an exact representation of a solar year, several days had accumulated since its institution. Therefore, Pope Gregory XIII decreed the elimination of 10 days from the year 1582. For many countries October 5, 1582 became October 15, 1582. However, different areas of the world did not adopt the calendar at the same time. Write a program to determine how many days to add to a Julian date to convert it to a Gregorian date. The following scale has been devised:

```
Add 10 days to Julian dates from 10/5/1582 to 2/28/1700
Add 11 days to Julian dates from  3/1/1700 to 2/28/1800
Add 12 days to Julian dates from  3/1/1800 to 2/28/1900
Add 13 days to Julian dates from  3/1/1900 to 2/28/2100
```

Input will be a Julian date of the format MM/DD/YYYY and output will be of the format ADD NN DAYS. Examples:

INPUT: Enter date: **09/02/1752** OUTPUT: **ADD 11 DAYS**

INPUT: Enter date: **03/01/1801** OUTPUT: **ADD 12 DAYS**

2.5 Sorting is the arranging of elements of a list into order based on the value of a particular field within the elements. There are several algorithms to accomplish the task of sorting including: bubble sort, shell sort, and quick sort. Each sorting algorithm's efficiency depends upon the number of elements to be sorted and the particular order in which they are originally given. The average number of comparisons needed to sort a list of N elements is given as:

```
N * (N-1) / 2                        for BUBBLE SORT
N * ((Log base 2 of N) squared)      for SHELL SORT
N * (Log base 2 of N)                for QUICK SORT
```

Assuming that the efficiency depends only on the number of elements in the list, N, determine the efficiency order of the different sorting algorithms and print them from most efficient to least efficient. N will be greater than 2. Example:

INPUT: Enter N: **128**

OUTPUT: **QUICK SORT**
SHELL SORT
BUBBLE SORT

2.6 During the summer and fall, NCNB has a golf league for employees. Write a program for a golfer to determine the status for each of the 9 holes, given the scores on each hole as input and using the pars listed below. The par is the score standard for each hole on a golf course. Also display the total number of strokes the golfer takes and the total par for the 9 holes. If a golfer shoots the standard score for a hole, the status is PAR. If on a hole a golfer shoots 1 score below the par, the status is BIRDIE. If the score is 2 below the par, the status is EAGLE. A score of 3 below the par is called a DOUBLE EAGLE. For scores 1 and 2 above par, the status is BOGEY and DOUBLE BOGEY respectively. Use the following pars for the holes:

```
HOLE: 1  2  3  4  5  6  7  8  9
------------------------------
PAR:  4  3  4  5  4  3  5  4  4
```

Example:

```
 INPUT: Enter score for hole 1: 3
        Enter score for hole 2: 3
        Enter score for hole 3: 5
        Enter score for hole 4: 7
        Enter score for hole 5: 6
        Enter score for hole 6: 1
        Enter score for hole 7: 2
        Enter score for hole 8: 3
        Enter score for hole 9: 4

OUTPUT: HOLE  PAR  SCORE  STATUS
        ----  ---  -----  ------
         1     4     3    BIRDIE
         2     3     3    PAR
         3     4     5    BOGEY
         4     5     7    DOUBLE BOGEY
         5     4     6    DOUBLE BOGEY
         6     3     1    EAGLE
         7     5     2    DOUBLE EAGLE
         8     4     3    BIRDIE
         9     4     4    PAR
              ---  -----
              36    34
```

2.7 In 45 B.C. Julius Caesar instituted the use of a calendar. The Julian calendar consists of a perpetual cycle of three years of 365 days followed by one year of 366 days. However, the exact solar year consists of 365 days, 5 hours, 48 minutes, 47.8 seconds. Thus, at the end of 1 Julian year, the Julian calendar is ahead of a solar calendar that will complete its year in another 5 hours 48 minutes and 47.8 seconds. Write a program to determine how many days, hours, minutes, and seconds the Julian calendar is behind/ahead of an imaginary calendar that is based on exact solar years. Assume that both calendars start at the same time and a comparison is done at the end of N Julian years, where N is input as a natural number less than 2000. Output must have 1 space between the number and its unit; 2 spaces between the unit and the next number. Seconds are displayed using the form ##.#. Examples:

INPUT: Enter N: **1**
OUTPUT: **0 DAYS 5 HOURS 48 MIN 47.8 SEC AHEAD**

INPUT: Enter N: **2**
OUTPUT: **0 DAYS 11 HOURS 37 MIN 35.6 SEC AHEAD**

INPUT: Enter N: **3**
OUTPUT: **0 DAYS 17 HOURS 26 MIN 23.4 SEC AHEAD**

INPUT: Enter N: **4**
OUTPUT: **0 DAYS 0 HOURS 44 MIN 48.8 SEC BEHIND**

INPUT: Enter N: **5**
OUTPUT: **0 DAYS 5 HOURS 3 MIN 59.0 SEC AHEAD**

2.8 In June 1989, Barb and Carolyn founded a suggestion committee for the NCNB Systems and Programming Division in Tampa. In August Joe was placed on the suggestion committee. In September Carolyn resigned and Doug took her place. The committee declared that a new person from a waiting list will replace a committee member after he/she has served for six months. With the following waiting list, write a program to display the 3 committee members for every month that someone is replaced. The names on a line must follow the logical order started in the example. Input will be the last month and year that the suggestion committee meets before it disassembles. The date input will not be later than 5, 1992.

Waiting list: JACKIE, TOM, LOVETTA, GREG, TONY, AL, KAREN, JAN, NORM, TRUDY, THERESA, ALICE, DAVE, JIM, STEVE

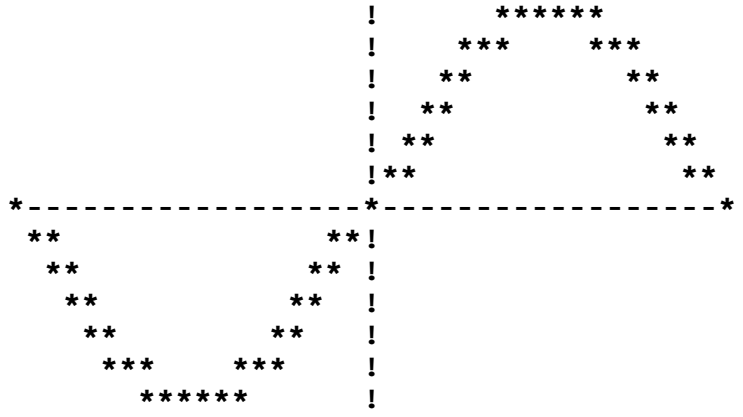
Example:

INPUT: Enter month, year: **6, 1990**

OUTPUT:
9/1989 - BARB JOE DOUG
12/1989 - JACKIE JOE DOUG
2/1990 - JACKIE TOM DOUG
3/1990 - JACKIE TOM LOVETTA
6/1990 - GREG TOM LOVETTA

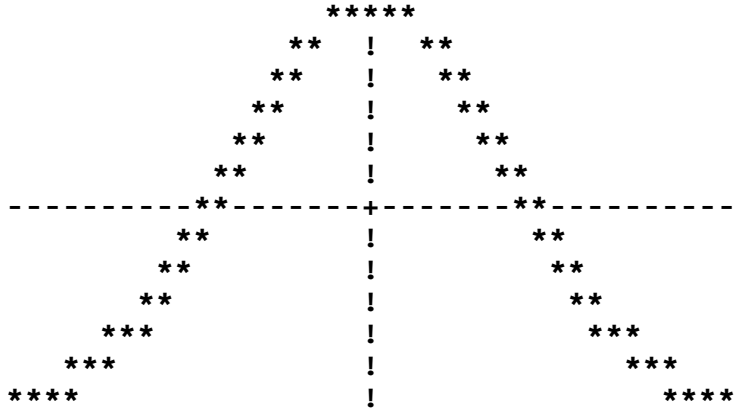
2.9 Write a program to graph both the sine function and the cosine function using asterisks. First, the program is to clear the screen. With dashes {-} for the x-axis and exclamation marks {!} for the y-axis, display the axes extending across the entire screen with the center (0,0) in the middle of the screen, represented by a {+}. With asterisks {*}, graph the sine function varying the x-coordinate from -PI to PI, where (-PI,0) is the left most {-} and (PI,0) is the right most {-}. The top most {!} is (0,1) and the bottom most {!} is (0,-1). Allow the user to press any key to clear the screen and display the axes again. This time graph the cosine function with the same coordinate dimensions. Allow the user to press any key to clear the screen. The graphs will look similar to the following. Example:

OUTPUT: (Screen clears and the axes is drawn before the graph is drawn from left to right. Graph will look similar to below, but it extends to the dimensions of the terminal.)



INPUT: (Press any key)

OUTPUT: (Screen clears and the axes is drawn before the graph is drawn from left to right- similar to below.)



INPUT: (Press any key) OUTPUT: (Screen clears)

2.10 A new employee of the NCNB computer programming department desires training. Write a program to produce the following NCNB menu of currently available in-house computer courses. Estimate the TOTAL number of hours he/she will spend on training given the courses that he/she has selected. Upon completion of course selection, 000-000 is entered. The screen is then cleared, and the course names chosen are displayed again in the order that they were selected. Estimated hours for each course selected must appear to the right of the course name: The heading "EST. HOURS" starts 26 columns after the first column of the "COURSE NAME," as shown in the menu selection. Also, the TOTAL estimated hours must show underneath all the individual course hours in the form: TOTAL = ##.# - ## HOURS. Example:

OUTPUT: **NCNB IN-HOUSE TRAINING LIST**

COURSE #	**COURSE NAME**	**EST. HOURS**
187-11X	**ISPF/PDS FUNDAMENTALS**	**6.5 - 8**
187-15X	**ISPF/PDS FOR PROGRAMMERS**	**4.5 - 6**
220-AXX	**JCL FUNDAMENTALS**	**15 - 20**
200-AXX	**VSAM CONCEPTS**	**4 - 7**
123-2XX	**MVS/SP/XA VSAM**	**7 - 11**
130-11X	**CICS/VS SKILLS I**	**6 - 8**
130-15X	**CICS/VS SKILLS II**	**4 - 6**

INPUT: Enter course # (or 000-000 to end): **220-AXX**
Enter course # (or 000-000 to end): **130-11X**
Enter course # (or 000-000 to end): **187-11X**
Enter course # (or 000-000 to end): **000-000**

OUTPUT: (Screen is cleared)

COURSE NAME	**EST. HOURS**
JCL FUNDAMENTALS	**15 - 20**
CICS/VS SKILLS I	**6 - 8**
ISPF/PDS FUNDAMENTALS	**6.5 - 8**
	TOTAL = 27.5 - 36 HOURS

3.1 A company would like to use an acronym as a phone number that is easy for the public to remember. They would like a word that can be used for the last several digits of their number, where each letter corresponds to a particular digit. Each possible word will be 4 or 5 letters long. Write a program to display all possible acronym phone numbers (in the order listed below from left to right, top to bottom) for an input number of the format XXX-XXXX. Assume that at least one word will satisfy the requirements given the following word list options:

```
AGENT  SOAP  MONEY  JEWEL  BALL   LOANS  CARE   SAVE   CALL
PAVE   KEEP  KINGS  KNIFE  KNOCK  JOINT  JUICE  LOBBY  RATE
```

Use the following letters to correspond to the digits 2 to 9:

```
A B C   D E F   G H I   J K L   M N O   P R S   T U V   W X Y
  2       3       4       5       6       7       8       9
```

Examples:

INPUT: Enter phone #: **555-3935** OUTPUT: **55J-EWEL**

INPUT: Enter phone #: **555-2255** OUTPUT: **555-BALL**
555-CALL

3.2 Write a program to select words given a string of letters with a wildcard, {*}. The following words make up the word list:

```
COMPUTE,    COMPUTER,   COMPUTERS, COMPORT,   COMPUTES,
COMPUTED,   ATTRACTIVE, ABRASIVE,  ADAPTIVE,  ACCEPTIVE,
AERATING,   CONTESTED,  CONTESTER, CORONETS,  CONTESTS,
CONTESTERS, COUNTESS,   CREATIVE,  CREATE,    CREATURE,
CREATION,   EVERYBODY,  EVERYONE,  EMPTY,     ELECTION
```

The wildcard could come before, after, or between the letters of the string. The program must display all words in the list (in order from left to right, top to bottom) separated by 2 spaces that are of the form input, where the wildcard could represent 0 to 10 letters. The words may be in any order. If no words are found then display the message NO WORDS FOUND. The program is to continue to display words and to accept as input a string with a wildcard until a string is entered without a wildcard. Example:

```
 INPUT: Enter string: CREAT*
OUTPUT: CREATIVE  CREATE  CREATURE  CREATION
 INPUT: Enter string: *TION          INPUT: Enter string: *ATER
OUTPUT: CREATION  ELECTION          OUTPUT: NO WORDS FOUND
 INPUT: Enter string: E*Y            INPUT: NO
OUTPUT: EVERYBODY  EMPTY            OUTPUT: (Program terminates)
```

3.3 Three local cross country teams compete in a double dual race. Each team consists of seven runners, but only the first five finishers of a team contribute to that team's score. As the runners cross the finish line, gasping for breath, the judge writes the INITIAL of the runner's team name and the NUMBER indicating the runner's finishing position, e.g. 1 for 1st, 2 for 2nd and so on. To find the score for teams A and B and to decide which of the two wins, the scorekeeper temporarily eliminates all of C's positions, and then repositions the runners from A and B into places 1 through 14. The team's score consists of the sum of the places of their first five runners. The lower team score wins. If there is a tie then the team whose sixth runner crossed the finish line first is the winner.

Write a program that computes the score for each pair of three teams and determines the winner of each pair (pairs may be displayed in any order). The program must allow the user to assign all 21 runners' team INITIAL to finishing places. Team initials can be any letter in the alphabet. Example:

INPUT:
Place 1: **A**
Place 2: **B**
Place 3: **A**
Place 4: **B**
Place 5: **A**
Place 6: **B**
Place 7: **C**
Place 8: **C**
Place 9: **C**
Place 10: **C**
Place 11: **B**
Place 12: **A**
Place 13: **C**
Place 14: **B**
Place 15: **C**
Place 16: **B**
Place 17: **A**
Place 18: **A**
Place 19: **C**
Place 20: **B**
Place 21: **A**

Example of repositioning teams A and B:

1: A
2: B
3: A
4: B
5: A
6: B
7: B
8: A
9: B
10: B
11: A
12: A
13: B
14: A

OUTPUT:
TEAM A: 28 POINTS
TEAM B: 28 POINTS
TEAM B WINS!

TEAM A: 25 POINTS
TEAM C: 31 POINTS
TEAM A WINS!

TEAM B: 24 POINTS
TEAM C: 31 POINTS
TEAM B WINS!

3.4 Within NCNB's Application Systems Division, there are many programming teams consisting of 2 to 8 programmers and 1 team leader. Suppose that NORM is a team leader in charge of 3 programmers: AL, DOUG, and JAN. Norm has been given 30 programs (numbered 1 through 30) to distribute amongst his employees. Al is given all programs divisible by X, and Doug is given all programs divisible by Y. Jan is given all programs divisible by Z, and Norm takes the rest of the programs that were not assigned to anyone and will work on some himself or he will redistribute some later. Given X, Y, Z as different input numbers between 2 and 10 inclusive, write a program to display those program numbers shared by all 3 of the programmers, those shared between only 2, and those assigned solely to 1 person. If no programs are assigned exclusively to one particular group, then display NONE instead of program numbers. The output must be in the same format as shown: eight lines, with numbers in increasing order. Examples:

```
 INPUT: Enter X, Y, Z: 2, 3, 5

OUTPUT: AL, DOUG, AND JAN = 30
        AL AND DOUG = 6 12 18 24
        AL AND JAN = 10 20
        DOUG AND JAN = 15
        AL = 2 4 8 14 16 22 26 28
        DOUG = 3 9 21 27
        JAN = 5 25
        NORM = 1 7 11 13 17 19 23 29


 INPUT: Enter X, Y, Z: 2, 6, 9

OUTPUT: AL, DOUG, AND JAN = 18
        AL AND DOUG = 6 12 24 30
        AL AND JAN = NONE
        DOUG AND JAN = NONE
        AL = 2 4 8 10 14 16 20 22 26 28
        DOUG = NONE
        JAN = 9 27
        NORM = 1 3 5 7 11 13 15 17 19 21 23 25 29
```

3.5 The numbers 1 through 8 and a blank are randomly placed in a 3 X 3 array on the screen. For example:

```
5  8  3
   4  1
6  2  7
```

Write a program so that when a key from 1 - 8 is pressed, the corresponding number on the screen slides horizontally or vertically to the adjacent blank location. If a key is pressed that is not horizontally or vertically adjacent to the blank, the computer does nothing. (For example: in the above diagram, only the 3, 1, and 7 can be moved.) The user can repeat this process of sliding numbers until the digit 9 is pressed, which causes the program to terminate.Example:

```
OUTPUT:  4  3  1
         2  8  5
            7  6

 INPUT: (Press 2)

OUTPUT:  4  3  1
            8  5
         2  7  6

 INPUT: (Press 8)

OUTPUT:  4  3  1
         8     5
         2  7  6

 INPUT: (Press 6)
OUTPUT: (No change)

 INPUT: (Press 9)
OUTPUT: (Program terminates)
```

3.6 Write a program to simulate a chess game between two players. Display a chess board with the white chess pieces at the bottom of the board (designated by a leading W), and the black pieces at the top of the board (designated by a leading B). The board is set up according to the international chess notation: a letter of the alphabet designates vertical columns, and a number designates horizontal rows. The board must be displayed as below. The computer allows WHITE to move first, and then allows BLACK to move. The computer continues to alternate moves until a king is captured. The White king is designated as WK and the black king is designated as BK. The user enters a valid move in the format L#-L# where L is one of the letters (A, B, C, D, E, F, G, H) and # is one of the digits (1, 2, 3, 4, 5, 6, 7, 8). The piece occupying the coordinate of the first L# is moved to the coordinate of the second L#. If a piece is moved onto the opponent's square, the opponent's piece is removed. If a piece takes over an opponent's king, then print CHECK MATE, CCCCC WON (where CCCCC is either WHITE or BLACK). Only valid moves will be entered, and neither castling nor the "en passant" move will be done. No "check" warning is given in this game. Example:

```
OUTPUT: BR1 BK1 BB1 BQ  BK  BB2 BK2 BR2  !  8
        BP1 BP2 BP3 BP4 BP5 BP6 BP7 BP8  !  7
                                         !  6
                                         !  5
                                         !  4
                                         !  3
        WP1 WP2 WP3 WP4 WP5 WP6 WP7 WP8  !  2
        WR1 WK1 WB1 WQ  WK  WB2 WK2 WR2  !  1
       ----------------------------------
         A   B   C   D   E   F   G   H

 INPUT: Enter white move: E2-E4

OUTPUT: BR1 BK1 BB1 BQ  BK  BB2 BK2 BR2  !  8
        BP1 BP2 BP3 BP4 BP5 BP6 BP7 BP8  !  7
                                         !  6
                                         !  5
                    WP5                  !  4
                                         !  3
        WP1 WP2 WP3 WP4     WP6 WP7 WP8  !  2
        WR1 WK1 WB1 WQ  WK  WB2 WK2 WR2  !  1
        ----------------------------------
         A   B   C   D   E   F   G   H

 INPUT: Enter black move: F7-F6

OUTPUT: (continued on next page)
```

(output continued)

```
    OUTPUT: BR1 BK1 BB1 BQ  BK  BB2 BK2 BR2  !  8
            BP1 BP2 BP3 BP4 BP5     BP7 BP8  !  7
                                BP6          !  6
                                             !  5
                            WP5              !  4
                                             !  3
            WP1 WP2 WP3 WP4     WP6 WP7 WP8  !  2
            WR1 WK1 WB1 WQ  WK  WB2 WK2 WR2  !  1
            ---------------------------------
             A   B   C   D   E   F   G   H

     INPUT: Enter white move: D1-H5

    OUTPUT: BR1 BK1 BB1 BQ  BK  BB2 BK2 BR2  !  8
            BP1 BP2 BP3 BP4 BP5     BP7 BP8  !  7
                                BP6          !  6
                                    WQ   !  5
                            WP5              !  4
                                             !  3
            WP1 WP2 WP3 WP4     WP6 WP7 WP8  !  2
            WR1 WK1 WB1     WK  WB2 WK2 WR2  !  1
            ---------------------------------
             A   B   C   D   E   F   G   H

     INPUT: Enter black move: B8-C6

    OUTPUT: BR1     BB1 BQ  BK  BB2 BK2 BR2  !  8
            BP1 BP2 BP3 BP4 BP5     BP7 BP8  !  7
                BK1             BP6          !  6
                                    WQ   !  5
                            WP5              !  4
                                             !  3
            WP1 WP2 WP3 WP4     WP6 WP7 WP8  !  2
            WR1 WK1 WB1     WK  WB2 WK2 WR2  !  1
            ---------------------------------
             A   B   C   D   E   F   G   H

     INPUT: Enter white move: H5-E8

    OUTPUT: BR1     BB1 BQ  WQ  BB2 BK2 BR2  !  8
            BP1 BP2 BP3 BP4 BP5     BP7 BP8  !  7
                BK1             BP6          !  6
                                             !  5
                            WP5              !  4
                                             !  3
            WP1 WP2 WP3 WP4     WP6 WP7 WP8  !  2
            WR1 WK1 WB1     WK  WB2 WK2 WR2  !  1
            ---------------------------------
             A   B   C   D   E   F   G   H

            CHECK MATE, WHITE WON
```

3.7 Write a program to determine the date of Easter and the date of Lent for a given year between 1970 and 2009 inclusive. Easter falls on the first Sunday following the arbitrary Paschal Full Moon, which does not necessarily coincide with a real or astronomical full moon. The Paschal Full Moon occurs on one of the following 19 dates corresponding to a key. The key is the remainder obtained by dividing the year by 19.

Key	Date	Key	Date
0	April 14	10	March 25
1	April 3	11	April 13
2	March 23	12	April 2
3	April 11	13	March 22
4	March 31	14	April 10
5	April 18	15	March 30
6	April 8	16	April 17
7	March 28	17	April 7
8	April 16	18	March 27
9	April 5		

Therefore, the key for the year 1990 is 14 (because 1990 / 19 = 104 remainder 14), and the Paschal Full Moon occurs on April 10. Since April 10, 1990 is a Tuesday, Easter Sunday is April 15. Note, if the Paschal Full Moon falls on a Sunday, Easter is the following Sunday. The earliest Easter can fall is March 23, and the latest is April 25. Lent begins on Ash Wednesday which comes 40 days before Easter, excluding Sundays (46 days including Sundays). Note, January 1, 1970 is a Thursday, and every year between 1970 and 2009 that is divisible by 4 is a leap year. Examples:

```
 INPUT: Enter year: 1970
OUTPUT: EASTER IS ON MARCH 29
        LENT IS ON FEBRUARY 11

 INPUT: Enter year: 1996
OUTPUT: EASTER IS ON APRIL 7
        LENT IS ON FEBRUARY 21

 INPUT: Enter year: 2000
OUTPUT: EASTER IS ON APRIL 23
        LENT IS ON MARCH 8
```

3.8 Write a program to keep score for a bowler. Input will be 10 frames of numbers. Output will be the scoring of each frame, as shown below in the example. The standard method of scoring will be used in this program.

In each frame the bowler has at most two chances to roll down all 10 pins. If the bowler does not knock down all the pins in that frame, then his score is added to the number of pins that he previously knocked down. (Note: the bowler's score starts out as 0).

For the first 9 frames, if the bowler knocks all 10 pins down on the 2nd roll (indicated by a /) then his score for that frame is his previous score + 10 + the number of pins that he knocks down on his next roll in the next frame. If he rolls all the pins down on his 1st ball of a frame (indicated by an X) then he gets 10 + the total number of pins that he knocks down on the next 2 rolls. A new frame is started after 2 balls are rolled or all ten pins are knocked down on the 1st ball.

In the 10th frame, the bowler is allowed at most three rolls, depending upon his first and second rolls. If he gets all the pins down on the 1st roll then his score will be his previous score plus 10 plus the number of pins he knocks down on his next 2 rolls. If he knocks all 10 pins down after 2 rolls, his score will be his previous score + 10 + the number of pins he knocks down on his next roll. Frame inputs are right justified and the score is left justified in each frame. Example:

```
INPUT: Enter frame 1: 7/
       Enter frame 2: X
       Enter frame 3: 62
       Enter frame 4: X
       Enter frame 5: X
       Enter frame 6: 8/
       Enter frame 7: 63
       Enter frame 8: X
       Enter frame 9: X
       Enter frame 10: X9-
```

(Note: 62 indicates that 6 pins were knocked down on 1st roll
and 2 pins were knocked down on 2nd roll;
/ indicates all remaining pins knocked down on 2nd roll;
X indicates 10 pins knocked down on the 1st roll;
- indicates 0 pins knocked down.)

```
OUTPUT: -1- -2- -3- -4- -5- -6- -7- -8- -9- -10-
        ---!---!---!---!---!---!---!---!---!---!
         7/!  X! 62!  X!  X! 8/! 63!  X!  X!X9-!
        20 !38 !46 !74 !94 !110!119!149!178!197!
        ----------------------------------------
```

3.9 Write a program to solve a system of N equations with N unknowns. The program will first accept a value for N between 2 and 4 inclusive. For each equation the program should accept the values for the N coefficients followed by the constant. Each given system of equations will have one unique solution whose values will be integers. A possible algorithm to be used to solve an NxN system of equations is shown to the right of the example input/output. Examples:

```
INPUT: Enter N: 3
Enter coefficients for row1
Co1: 2
Co2: 4
Co3: 6
Enter constant: 4
Enter coefficients for row2
Co1: 2
Co2: 1
Co3: 1
Enter constant: 3
Enter coefficients for row3
Co1: -1
Co2: 2
Co3: 1
Enter constant: 2

OUTPUT: (1, 2, -1)
```

```
 2x + 4y + 6z = 4    (Divide by 2)
 2x +  y +  z = 3
-x + 2y +  z = 2

 1  2   3   2
 2  1   1   3   (Subtract 2*Row1)
-1  2   1   2

 1  2   3   2
 0 -3  -5  -1
-1  2   1   2   (Subtract -1*Row1)

 1  2   3   2
 0 -3  -5  -1   (Divide by -3)
 0  4   4   4

 1  2   3   2
 0  1  5/3 1/3
 0  4   4   4   (Subtract 4*Row2)

 1  2   3   2
 0  1  5/3 1/3
 0  0 -8/3 8/3  (Divide by -8/3)

 1  2   3   2
 0  1  5/3 1/3  (Subtract 5/3*Row3)
 0  0   1  -1

 1  2   3   2   (Subtract 3*Row3)
 0  1   0   2
 0  0   1  -1

 1  2   0   5   (Subtract 2*Row2)
 0  1   0   2
 0  0   1  -1

 1  0   0   1
 0  1   0   2
 0  0   1  -1
```

```
INPUT: Enter N: 2
Enter coefficients for row1
Co1: 1
Co2: 1
Enter constant: 1
Enter coefficients for row2
Co1: 2
Co2: 3
Enter constant: 6

OUTPUT: (-3, 4)
```

3.10 Write a program to solve cryptorithms (a puzzle in which the letters of the alphabet are written in a computational format). The object is to determine which digit corresponds to each letter in the puzzle to satisfy the computation. Input will be two 2-letter addends and a 3-letter sum. The program will generate a numeric solution for each of the letters used. The judging criteria will be such that there will always be at least one solution. Only the letters A, B, C, D, and E may be used in the input, and the leading letter may not be 0. Example:

INPUT: Enter first addend: **AB**
Enter second addend: **BC**
Enter sum: **DDB**

OUTPUT: (Only one of the following solutions must be shown)

A = 2		**3**		**4**		**5**		**6**		**7**		**8**	
B = 9	or	**8**	or	**7**	or	**6**	or	**5**	or	**4**	or	**3**	
C = 0		**0**		**0**		**0**		**0**		**0**		**0**	
D = 1		**1**		**1**		**1**		**1**		**1**		**1**	

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '91

1.1 Write a program to clear the screen and display the following:

```
COMPUTER CONTEST 1991
O                   9
M                   9
P                   1
U
T                   T
E                   S
R                   E
                    T
C                   N
O                   O
N                   C
T
E                   R
S                   E
T                   T
                    U
1                   P
9                   M
9                   O
1991 TSETNOC RETUPMOC
```

1.2 Write a program to display two random integers and their sum. Output must be displayed in the form: X + Y = ZZ, where X and Y are random integers between -9 and 9 inclusive. At least one space must appear between symbols and numbers. Output examples:

3 + -9 = -6 or **-5 + -2 = -7** or **7 + 8 = 15**

1.3 Since 1980, the only high school team to complete all 30 programming problems at the Florida High School Computer Contest was Coral Springs High School in 1984, finishing with a time of 2 hours and 21 minutes. As a result of finishing 10 one-point programs, 10 two-point programs, and 10 three-point programs, they had a maximum team score of 60 points: (10 x 1 + 10 x 2 + 10 x 3 = 60). Write a program to determine the team score for a school, given the team name and the number of 1-point, 2-point, and 3-point programs completed. Example:

```
INPUT: Enter team name: CORAL SPRINGS HS
  Enter # of 1 point programs: 7
  Enter # of 2 point programs: 3
  Enter # of 3 point programs: 1

OUTPUT: CORAL SPRINGS HS SCORED 16 POINTS
```

1.4 Write a program to display the screen format for a spreadsheet. The letters A through T are to appear on the first line, each separated by 1 space. The numbers 1 through 20 are to appear on the left most part of the next 20 lines. (Note: the letter A starts in the 4th column) Example:

```
   A B C D E F G H I J K L M N O P Q R S T
 1
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
```

1.5 Write a program to determine the number of teams participating in the Florida High School Computer Contest. The number of students participating will be given as input. Assume that every team has 4 students. Example:

```
 INPUT: Enter number of students: 168
OUTPUT: 42 TEAMS
```

1.6 Write a program to accept a word with distinct letters, and then accept a letter in the word. The program must then print the word vertically and horizontally, intersecting at the chosen letter. Example:

```
 INPUT: Enter word: COMPUTER
        Enter letter: M

OUTPUT:   C
          O
        COMPUTER
          P
          U
          T
          E
          R
```

1.7 A 19 digit account key identifies specific Dealer Floor Plan information. Each set of digits represents a particular number, as follows:

```
Digits  1 - 3  = Organization number
Digits  4 - 6  = Branch number
Digits  7 - 10 = Dealer number
Digits 11 - 13 = Class number
Digits 14 - 19 = Unit serial number
```

Write a program to display the five fields followed by the actual number taken from the digits in an input 19 digit account key. Example:

```
 INPUT: Enter account key: 0010071234001228334

OUTPUT: ORGANIZATION 001
        BRANCH 007
        DEALER 1234
        CLASS 001
        UNIT 228334
```

1.8 JCL (Job Control Language) allows a programmer to communicate with a computer operating system. The operating system maintains control over what the computer does. A job is a unit of work that the computer is to do. A job stream consists of JCL statements. The three main JCL statements, necessary to every job, consist of the following characters: JOB, EXEC, and DD. The JOB statement must be the first statement in a job stream, and it identifies the job to the system. The EXEC statement identifies the program (or procedure) to execute. The DD (data definition) statement describes the data used by the program. A JOB STEP consists of an EXEC statement and its DD statement(s) that follow. A job may have any number of job steps. Write a program to determine the number of job steps that are in a job stream, given several lines of partial JCL code (JOB, EXEC, or DD). The last input line will be two slashes //, which marks the end of the job stream. Example:

```
 INPUT: Enter line: JOB
        Enter line: EXEC
        Enter line: DD
        Enter line: EXEC
        Enter line: DD
        Enter line: DD
        Enter line: DD
        Enter line: EXEC
        Enter line: DD
        Enter line: //

OUTPUT: 3 JOB STEPS
```

1.9 Write a program to accept a one-line sentence and to replace all occurrences of the word MAN with the word PERSON and to replace all occurrences of the word MEN with PERSONS. The words MAN and MEN may be imbedded in other words. The output will be less than 80 characters in length. Examples:

INPUT: Enter sentence:
MEN AND WOMEN HAVE ENGAGED IN MENTAL MANIPULATIONS.

OUTPUT:
PERSONS AND WOPERSONS HAVE ENGAGED IN PERSONSTAL PERSONIPULATIONS.

1.10 Write a program to determine the winner of two teams participating in the FLORIDA HIGH SCHOOL COMPUTER CONTEST. The team with the most amount of points wins. In the event of a tie of points, the team with the lowest overall time is declared the winner. The overall time consists of the sum of the completion time of the last program finished plus 5 minutes for every penalty point accumulated. Input will be a team name, then its points, time (in the form HMM where H is hours and MM is minutes), and number of penalties. Examples:

INPUT: Enter team name: **TARAVELLA**
Enter points, time, penalties: **32, 155, 6**
Enter team name: **CORAL SPRINGS**
Enter points, time, penalties: **32, 212, 1**

OUTPUT: **CORAL SPRINGS WINS**

INPUT: Enter team name: **TARAVELLA**
Enter points, time, penalties: **33, 155, 6**
Enter team name: **CORAL SPRINGS**
Enter points, time, penalties: **32, 212, 1**

OUTPUT: **TARAVELLA WINS**

2.1 Write a program to produce a pyramid of N numbers where N is input as 6, 10, 15, 21, 28, 36, or 45. The format is shown below. Each single digit number has a leading zero, and each succeeding row has one more number than the previous row. Each number is separated by two spaces. Example:

INPUT: Enter N: **21**

OUTPUT:

```
            01
          02  03
        04  05  06
      07  08  09  10
    11  12  13  14  15
  16  17  18  19  20  21
```

2.2 Write a program to accept 5 decimal numbers and to display them lined up vertically with the decimal point in the same column. No more than 4 digits will be input on the right or the left of the decimal point. Display 9 dashes under the last number (4 dashes on the right side of the decimal column and 4 dashes on the left side). Display the sum of all 5 numbers under the dashes with its decimal point lined up with the other decimal points. Example:

INPUT: Enter #: **1.1234**
Enter #: **3.456**
Enter #: **123.45**
Enter #: **23.5**
Enter #: **678.9166**

OUTPUT:

```
   1.1234
   3.456
 123.45
  23.5
 678.9166
---------
 830.4460
```

2.3 Write a program to convert a partial BASIC program statement into a proper COBOL statement. The COBOL statement will be identical to the BASIC statement except that the relational symbols will be replaced by words:

>	IS GREATER THAN
<	IS LESS THAN
=	IS EQUAL TO
<= or =<	IS NOT GREATER THAN
>= or =>	IS NOT LESS THAN
<> or ><	IS NOT EQUAL TO

Example:

INPUT: Enter statement: **IF A < B OR C >= D**

OUTPUT: **IF A IS LESS THAN B OR C IS NOT LESS THAN D**

2.4 Write a program to rank N teams in a league. Input will be a set of team names each followed by the number of wins and losses on the next line. Output will be the teams in the order of number of wins, with its rank in the league appearing to the left of the name. Each differing place is separated by 1 blank line. If more than 1 team is tied, the teams are displayed in alphabetical order with no blank lines separating them. The number of wins is displayed 16 columns after the column with the team rank. Example:

```
 INPUT: Enter N: 6
        Enter team: STRIKES
        Enter wins, losses: 29, 43
        Enter team: PAR FOUR
        Enter wins, losses: 35, 37
        Enter team: WINNERS
        Enter wins, losses: 35, 37
        Enter team: RUNNERS
        Enter wins, losses: 35, 37
        Enter team: FORCE
        Enter wins, losses: 29, 43
        Enter team: HIGH ROLLERS
        Enter wins, losses: 42, 30

OUTPUT: 1 HIGH ROLLERS  42 , 30

        2 PAR FOUR      35 , 37
        2 RUNNERS       35 , 37
        2 WINNERS       35 , 37

        5 FORCE         29 , 43
        5 STRIKES       29 , 43
```

2.5 Write a program to ALWAYS guess a user's secret number within 7 guesses, where the number is be between 1 and 127 inclusive. The user will respond either H--Higher, L--Lower, or R--Right on. Have the program display guesses of this form: GUESS X: YYY, where X is the guess # (1 - 7), and YYY is the number guessed. Example:

```
OUTPUT: GUESS 1: 64
 INPUT: Enter H, L, or R: L
OUTPUT: GUESS 2: 32
 INPUT: Enter H, L, or R: H
OUTPUT: GUESS 3: 48
 INPUT: Enter H, L, or R: L
OUTPUT: GUESS 4: 40
 INPUT: Enter H, L, or R: H
OUTPUT: GUESS 5: 44
 INPUT: Enter H, L, or R: L
OUTPUT: GUESS 6: 42
 INPUT: Enter H, L, or R: H
OUTPUT: GUESS 5: 43
 INPUT: Enter H, L, or R: R
OUTPUT: (program terminates)
```

2.6 Write a program to display text in pyramid form. The first word is displayed centered on the top, with respect to column 20 of your screen. Each succeeding line contains the least amount of words that causes it to exceed the previous line by at least 2 characters. Each line is then centered with respect to column 20. If a line contains an even amount of characters, then center the line in such a way that one extra character appears on the left half of column 20 as opposed to the right half. The last remaining line is to be centered with respect to column 20 regardless of its length. All words are separated by 1 space. No more than 127 characters will be entered. Example:

INPUT: Enter text: **WRITE A PROGRAM TO DISPLAY TEXT IN PYRAMID FORM. THE FIRST WORD IS DISPLAYED CENTERED ON THE TOP.**

OUTPUT:

WRITE
A PROGRAM
TO DISPLAY TEXT
IN PYRAMID FORM. THE
FIRST WORD IS DISPLAYED
CENTERED ON THE TOP.

2.7 Write a program to display a rectangle of asterisks centered on the screen. The length and width are input. Example:

INPUT: Enter length, width: **7, 5**

OUTPUT: (the following rectangle is centered on the screen)

```
*******
*     *
*     *
*     *
*******
```

2.8 Write a program to display a bar graph for the lengths of the computer contests from 1980-1991. The title of the graph is input and then displayed indented 4 characters on the first line. Twelve numbers (less than 32768) are input corresponding to the length of contests 1980, 1981, ... 1991. Output will be a bar graph with a maximum of 20 vertical asterisks appearing for the year with the largest number. Each vertical asterisk represents the value of the maximum length divided by 20. This increment value for each asterisk is displayed on the first line after the title. Three spaces separate the title and the phrase "ASTERISK = ####.##". The amount of asterisks for the other years is determined by dividing the length by the increment value and truncating the result. Example:

```
INPUT: Enter title: BASIC PROGRAMS
       Enter # for 1980: 5475
       Enter # for 1981: 5640
       Enter # for 1982: 13848
       Enter # for 1983: 10346
       Enter # for 1984: 10932
       Enter # for 1985: 18478
       Enter # for 1986: 20093
       Enter # for 1987: 19034
       Enter # for 1988: 19140
       Enter # for 1989: 20840
       Enter # for 1990: 19459
       Enter # for 1991: 17014

OUTPUT:    BASIC PROGRAMS   ASTERISK = 1042.00
       20                                  *
       19                        *         *
       18                        *   *   *   *   *
       17                    *   *   *   *   *   *
       16                    *   *   *   *   *   *   *
       15                    *   *   *   *   *   *   *
       14                    *   *   *   *   *   *   *
       13        *           *   *   *   *   *   *   *
       12        *           *   *   *   *   *   *   *
       11        *           *   *   *   *   *   *   *
       10        *       *   *   *   *   *   *   *   *
        9        *   *   *   *   *   *   *   *   *   *
        8        *   *   *   *   *   *   *   *   *   *
        7        *   *   *   *   *   *   *   *   *   *
        6        *   *   *   *   *   *   *   *   *   *
        5  *   *   *   *   *   *   *   *   *   *   *   *
        4  *   *   *   *   *   *   *   *   *   *   *   *
        3  *   *   *   *   *   *   *   *   *   *   *   *
        2  *   *   *   *   *   *   *   *   *   *   *   *
        1  *   *   *   *   *   *   *   *   *   *   *   *
       ---------------------------------------------
          80  81  82  83  84  85  86  87  88  89  90  91
```

2.9 Write a program to produce a file maintenance list for a store. Each day customers come into the store to either purchase, exchange, or return an item. The store assigns to each customer a unique customer ID, consisting of 2 letters followed by 2 digits. If the customer purchases an item, then the customer ID and the item name (A,B,...Y, or Z) is entered in the file. The file is duplicated each night. Customers may exchange or return an item on the next day. If a customer exchanges an item, the item name is changed in the file corresponding to the customer. If a customer returns an item, his record entry is deleted from the file. Both files are sorted alpha-numerically according to the ID. Given the contents of yesterday's file and the contents of today's file, display the records that were added, changed, and deleted, under the following headings: ADDED, CHANGED, DELETED. Display the records in order of ID within each section. Each record is to be displayed with a space between the ID and the item. If the record was changed, display the old contents followed by a space and the new contents. Each section is to be separated by a blank line. Display the total number of each function at the end. Example:

```
 INPUT: Enter # of entries in yesterday's file: 4
        Enter ID: AB12
        Enter item: C
        Enter ID: CH39
        Enter item: R
        Enter ID: CH40
        Enter item: D
        Enter ID: CR11
        Enter item: A

 INPUT: Enter # of entries in today's file: 3
        Enter ID: AB12
        Enter item: C
        Enter ID: CH40
        Enter item: T
        Enter ID: CR13
        Enter item: A

OUTPUT: ADDED
        CR13 A

        CHANGED
        CH40 D T

        DELETED
        CH39 R
        CR11 A

        TOTAL ADDED = 1
        TOTAL CHANGED = 1
        TOTAL DELETED = 2
```

2.10 Write a program to display the contents of one of Doug Woolley's computer diskettes for the Florida High School Computer Contest, given the year of the contest. The contests from 1983-1991 have had 10 1-point problems, 10 2-point problems, and 10 3-point problems. In 1980, there were 10 1's, 10 2's, and 12 3's. In 1981, there were 5 1's, 5 2's, and 5 3's. In 1982, there were 10 1's, 12 2's, and 8 3's. Each year has its own disk. For each year, there is one BASIC program for each problem and one Pascal program for each problem. The names are of the form:

XXX#TYY.ZZZ

where XXX is ONE, TWO or THR; # is a problem number (1 through 5, 8, 10, or 12 depending on the year); T is T for test; YY is year of contest (80 through 91); ZZZ is BAS or PAS for BASIC or Pascal.

In addition, each diskette has 12 files of the form:

FHSYY-#.ZZZ

where FHS signifies Florida High School, YY is the year (80 through 91); # is the problem number (1, 2, or 3); ZZZ is the extension (PRB, JDG, PG1, or PG2), representing a compilation of problems, judging criteria, BASIC programs, and Pascal programs respectively. Given a year as input, display 20 files at a time, waiting for a key press before displaying the next 20. The files are to be grouped primarily by their extension in the following order: PRB, JDG, PG1, PG2, BAS, PAS. The files are then grouped secondarily within some of these sub-groups by prefix order: ONE, TWO, THR. Every sub-group is arranged in order of problem numbers. Example:

INPUT: Enter year: **1991**

OUTPUT: **FHS91-1.PRB**
FHS91-2.PRB
FHS91-3.PRB
FHS91-1.JDG
FHS91-2.JDG
FHS91-3.JDG
FHS91-1.PG1
FHS91-2.PG1
FHS91-3.PG1
FHS91-1.PG2
FHS91-2.PG2
FHS91-3.PG2
ONE1T91.BAS
ONE2T91.BAS
ONE3T91.BAS
ONE4T91.BAS
ONE5T91.BAS
ONE6T91.BAS
ONE7T91.BAS
ONE8T91.BAS
INPUT: (press any key)
OUTPUT: (continued on next page)

```
OUTPUT: ONE9T91.BAS
        ONE10T91.BAS
        TWO1T91.BAS
        TWO2T91.BAS
        TWO3T91.BAS
        TWO4T91.BAS
        TWO5T91.BAS
        TWO6T91.BAS
        TWO7T91.BAS
        TWO8T91.BAS
        TWO9T91.BAS
        TWO10T91.BAS
        THR1T91.BAS
        THR2T91.BAS
        THR3T91.BAS
        THR4T91.BAS
        THR5T91.BAS
        THR6T91.BAS
        THR7T91.BAS
        THR8T91.BAS
 INPUT: (press any key)
OUTPUT: THR9T91.BAS
        THR10T91.BAS
        ONE1T91.PAS
        ONE2T91.PAS
        ONE3T91.PAS
        ONE4T91.PAS
        ONE5T91.PAS
        ONE6T91.PAS
        ONE7T91.PAS
        ONE8T91.PAS
        ONE9T91.PAS
        ONE10T91.PAS
        TWO1T91.PAS
        TWO2T91.PAS
        TWO3T91.PAS
        TWO4T91.PAS
        TWO5T91.PAS
        TWO6T91.PAS
        TWO7T91.PAS
        TWO8T91.PAS
 INPUT: (press any key)
OUTPUT: TWO9T91.PAS
        TWO10T91.PAS
        THR1T91.PAS
        THR2T91.PAS
        THR3T91.PAS
        THR4T91.PAS
        THR5T91.PAS
        THR6T91.PAS
        THR7T91.PAS
        THR8T91.PAS
        THR9T91.PAS
        THR10T91.PAS
```

3.1 Write a program to simulate a baseball game of 9 innings. The standard baseball rules apply, but the bottom of the 9th inning is always played. Pitchers randomly throw strikes 40% of the time and the batters never swing at the ball. If 4 balls are thrown before 3 strikes are thrown, the batter walks to first base. When 4 batters from one team walk in one inning, 1 run is earned. Each batter that walks thereafter in the same inning earns a run for the team. 3 strikes make 1 out, and after 3 outs the next team bats. Because the program is random, executions will differ slightly. Examples:

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**SCORE**
TEAM A !	0	0	0	3	0	3	0	0	0 !	6
TEAM B !	1	0	0	2	0	0	0	2	0 !	5

TOTAL # OF STRIKES: 235
TOTAL # OF BALLS: 343
TOTAL # OF WALKS: 77
TOTAL # OF STRIKE OUTS: 54

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**SCORE**
TEAM A !	0	1	0	0	1	1	0	0	0 !	3
TEAM B !	0	0	2	0	0	2	2	0	3 !	9

TOTAL # OF STRIKES: 251
TOTAL # OF BALLS: 385
TOTAL # OF WALKS: 88
TOTAL # OF STRIKE OUTS: 54

3.2 Write a program to display the units digit of the following expression:

A ^ X + B ^ Y + C ^ Z

where A, B, C are input as numbers between 1 and 15 inclusive, and X, Y, Z are input as numbers between 100 and 999 inclusive.
The symbol ^ means "raised to the power of". Examples:

INPUT: Enter A, X: **3, 997**
Enter B, Y: **7, 998**
Enter C, Z: **13, 999**

OUTPUT: **9**

INPUT: Enter A, X: **5, 103**
Enter B, Y: **4, 102**
Enter C, Z: **3, 101**

OUTPUT: **4**

3.3 Write a program to display all the digits of the result of X raised to the Y power, where X and Y are positive integers less than 100. The result will not exceed 200 digits, but it may wrap around the screen. Examples:

```
 INPUT: Enter X, Y: 2, 99
OUTPUT: 633825300114114700748351602688


 INPUT: Enter X, Y: 45, 67
OUTPUT: 582422873843435732243403365900823690091753194215381
        550018222590908115610802697759140755806583911180496
        2158203125
```

3.4 Write a program to assign user LOGON ID's to several people who would like access to their company's mainframe computer. Input will be several lines of full names (entered one line at a time) and followed by a line with the word END to indicate the end of input. Each input line will be of the format:

(first name)(1 space)(middle name or initial or nothing)
(1 space)(last name)

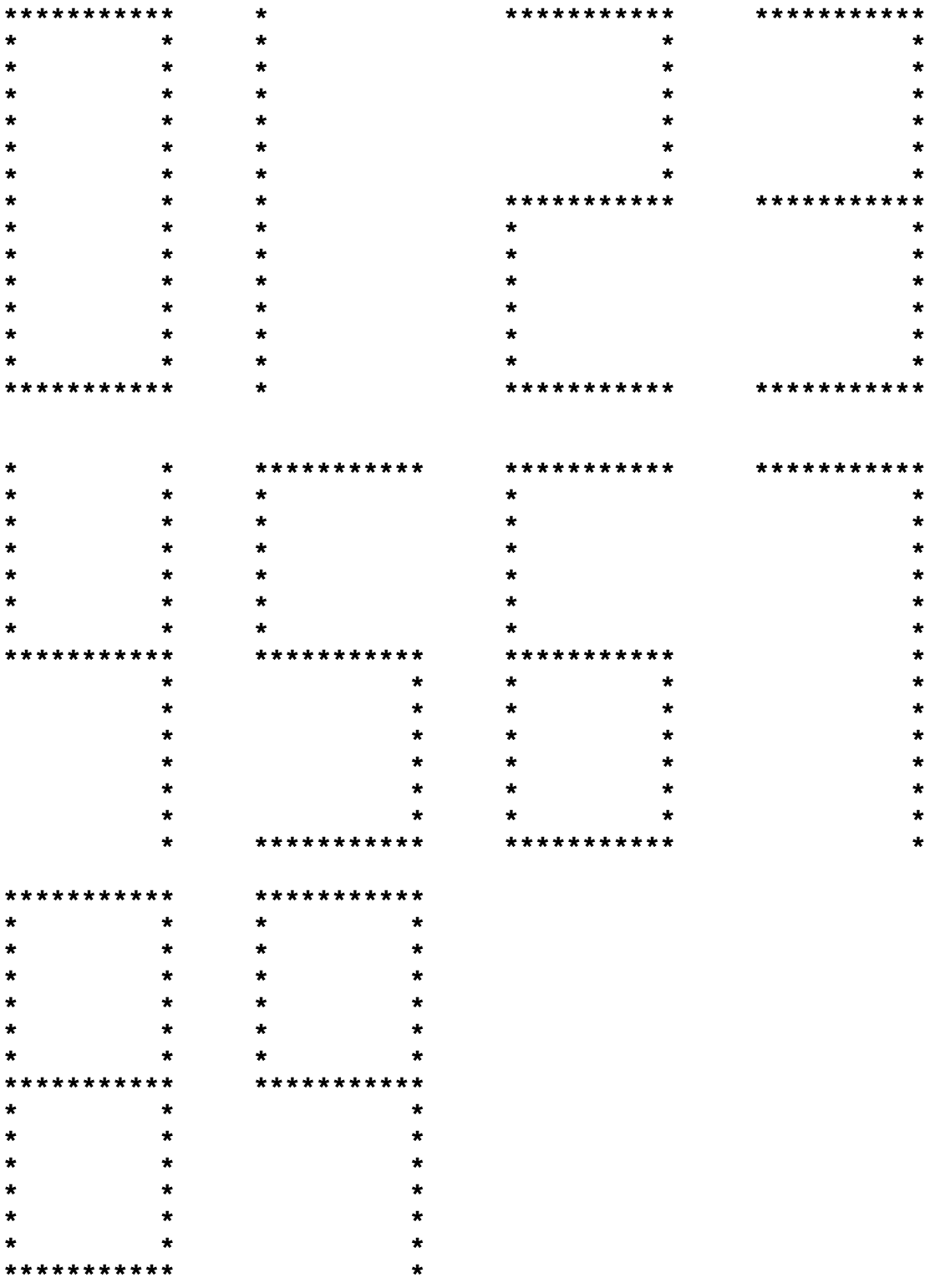
If no middle name or initial is entered then only 1 space separates the first and last names. Output will consist of each full name followed by his/her LOGON ID. All IDs are to be lined up vertically, starting 19 columns after the first character in the name. The LOGON ID is generally of the form:

SD(3 initials of first, middle, last name)1

There are two exceptions. If no middle initial or name is entered, then the letter X is used as the middle initial. If more than one person has the same initials then the common names are sorted alphabetically by last name then first name. The first sorted name is assigned a middle initial of 1; the next name alphabetically is assigned a middle initial of 2; and so on. Example:

```
 INPUT: Enter name: DOUG E WOOLLEY
        Enter name: DAVE E WEAVER
        Enter name: ALICE DOCK
        Enter name: ANNE VINCENT
        Enter name: AL DATSON
        Enter name: DON ENGLAND CHANG
        Enter name: END

OUTPUT: DOUG E WOOLLEY     SDD2W1
        DAVE E WEAVER      SDD1W1
        ALICE DOCK         SDA2D1
        ANNE VINCENT       SDAXV1
        AL DATSON          SDA1D1
        DON ENGLAND CHANG  SDDEC1
```

3.5 Write a program to display the numbers 0 through 9, one at a time in enlarged block format (15 characters high by 11 characters wide). Each number must be displayed for approximately 1 second before clearing the screen and displaying the next number. The following are examples of the numbers:



3.6 Write a program to evaluate an arithmetic expression using positive integers less than 10 and the symbols + and -. Several pairs of parenthesis may also be used. Examples:

INPUT: Enter expression: **(8+4)-(8-(6+1))**

OUTPUT: **11**

INPUT: Enter expression: **7-9+(1-(6+2))-5+9**

OUTPUT: **-5**

INPUT: Enter expression: **((1+(1+2-3+4)-2+5)-3)**

OUTPUT: **5**

3.7 Write a program to determine the 2 dates that an employee is paid in a given month in 1991. Normally an employee is paid on the 15th and the last day of the month except if this date falls on a weekend or on a holiday. In such a case, the pay day is the first working day proceeding the normally expected pay day. Employees work Monday through Friday except on holidays. The program must first accept all the holidays for that year, input as MM, DD. After entering 0,0 to terminate holiday entries, the program accepts the month number for the pay days desired. Output must be the 2 pay days in that month in the form: day of week; month name; day. The program is to continue accepting month numbers and displaying pay days until 0 is entered for the month number. January 1, 1991 was a Tuesday. Example:

INPUT: Enter holiday MM, DD: **1, 1**
Enter holiday MM, DD: **1, 21**
Enter holiday MM, DD: **2, 28**
Enter holiday MM, DD: **3, 29**
Enter holiday MM, DD: **0, 0**

INPUT: Enter month #: **1**

OUTPUT: **TUESDAY JANUARY 15**
THURSDAY JANUARY 31

INPUT: Enter month #: **3**

OUTPUT: **FRIDAY MARCH 15**
THURSDAY MARCH 28

INPUT: Enter month #: **0**

OUTPUT: (program terminates)

3.8 Write a program to display all 3 x 3 magic squares (consisting of the digits 1 - 9) that contain a given digit in a designated position.

```
6  7  2      is a 3 x 3 magic square where the sum of
1  5  9      all the elements in each row, column, and
8  3  4      diagonal equal the same number, 15.
```

In this example, the digit 7 is in row 1, column 2, designated as (1,2). Given one digit as input (not 5) and the row and column this digit must appear, display all 3 x 3 magic squares that have that digit in the designated position. If no magic squares can be formed then display "NO SOLUTION". Hint: If a solution is possible, there will be two magic squares, each of which can be formed by rotating the outer numbers of a valid magic square and by interchanging the elements on each side of the 2nd row, 2nd column, or a diagonal (depending on the row and column of the number designated). The two magic squares may appear in either order. Examples:

```
 INPUT: Enter digit: 9
        Enter row, col: 1, 2

OUTPUT: 2  9  4
        7  5  3
        6  1  8

        4  9  2
        3  5  7
        8  1  6
```

```
 INPUT: Enter digit: 4
        Enter row, col: 3, 3

OUTPUT: 6  7  2
        1  5  9
        8  3  4

        6  1  8
        7  5  3
        2  9  4
```

```
 INPUT: Enter digit: 4
        Enter row, col: 1, 2

OUTPUT: NO SOLUTION
```

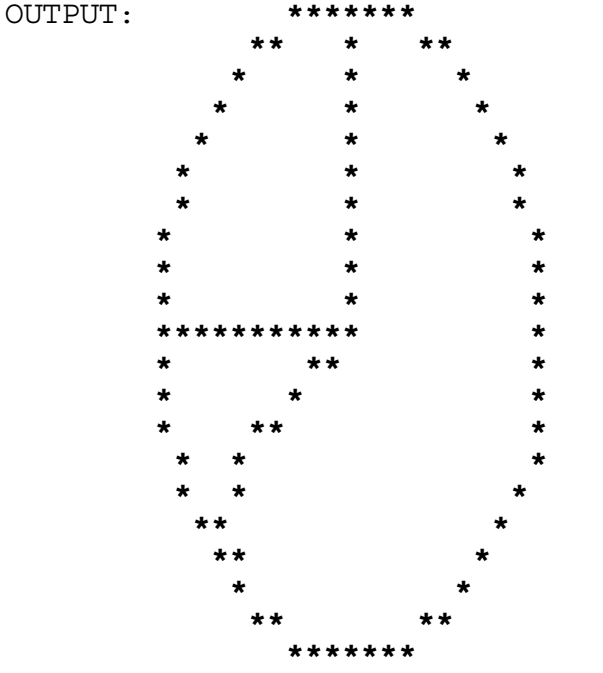
3.9 Write a program to display a pie graph on the screen using asterisks and the letters A, D, and N.

Input will be 3 percentages to divide the circle corresponding to 3 options on a survey: Agree, Disagree, or Neutral.

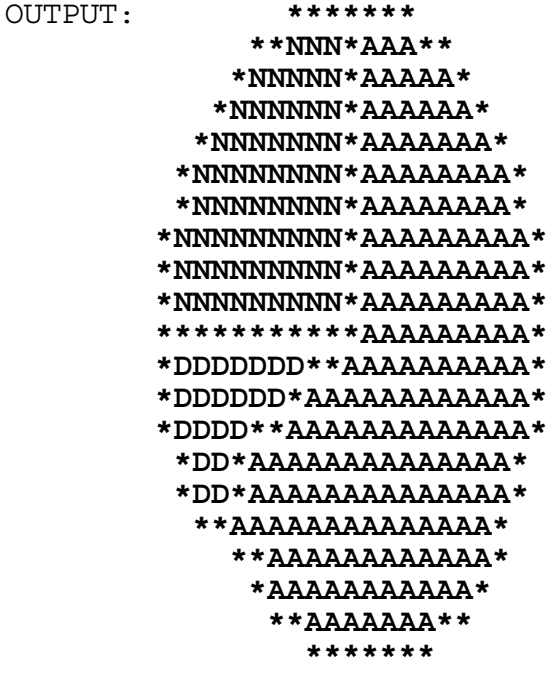
Output will be a circle of radius 10 characters. The circle is then partitioned by 3 line segments of asterisks stemming from the center. The first percentage entered (those that Agree) is represented by a proportional region of the circle enclosed by two segments: One segment is drawn from the center to the top of the circle, and another segment is drawn from the center to a point on the perimeter, enclosing a clock-wise region. Another segment is drawn from the center to the perimeter so that the next area clock-wise represents the percentage that disagrees. The third region clock-wise represents the percentage that are neutral. After the user presses a key, the 3 regions are filled with either A's, D's, or N's, corresponding to its region. Although your output should look very similar to the judging criteria, minor variations will be accepted. After pressing a key to fill the regions, all regions must be at least 90% filled. No letters may replace any of the asterisks. Example:

INPUT: Enter 3 percentages: **64, 11, 25**

OUTPUT:



OUTPUT:



INPUT: (press any key)

3.10 Write a program to convert a base M numeral into a base N numeral where M and N are entered as 2, 4, 8, or 16. Both numerals may have as many as 64 digits. Output must not contain leading zeroes. Examples:

INPUT: Enter numeral: **123456789ABCDEF0123456789ABCDEF0**
Enter base M: **16**
Enter base N: **4**

OUTPUT:
10203101112132021222330313233000102031011121320212223303132330 0

INPUT: Enter numeral:
12345670123456701234567012345670123456701
Enter base M: **8**
Enter base N: **16**

OUTPUT: **14E5DC14E5DC14E5DC14E5DC14E5DC1**

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '92

1.1 Write a program to display the following company name:

```
GGGGG   TTTTT   EEEEE
G         T     E
G GGG     T     EEEEE   DATA SERVICES
G   G     T     E
GGGGG     T     EEEEE
```

1.2 GTE became a corporate entity in 1918 in Wisconsin and was named: RICHLAND CENTER TELEPHONE COMPANY. Over the years this company has taken on various names as a result of its mergers and to reflect its current focus:

1920	COMMONWEALTH TELEPHONE COMPANY
1926	ASSOCIATED TELEPHONE UTILITIES COMPANY
1935	GENERAL TELEPHONE CORPORATION
1959	GENERAL TELEPHONE & ELECTRONICS CORPORATION
1982	GTE CORPORATION

Write a program to print the name of the organization in a given year from 1918 to 1992. Examples:

```
 INPUT: Enter year: 1950
OUTPUT: GENERAL TELEPHONE CORPORATION

 INPUT: Enter year: 1982
OUTPUT: GTE CORPORATION
```

1.3 In 1990, GTE ranked 13th in the Forbes Super 50 1990 list of companies. The list ranks the biggest U.S. companies on a composite of revenue, net income, assets and market value. The following year GTE ranked 7th in the Forbes Super 50 1991 list of companies, rising up 6 places. Write a program to display a company's projected 1992 ranking after accepting a company's rank for 1991 and the number of places the company rises the following year. Examples:

```
 INPUT: Enter 1991 rank: 40
        Enter number of places: 8

OUTPUT: 32


 INPUT: Enter 1991 rank: 21
        Enter number of places: 13

OUTPUT: 8
```

1.4 GTE is the largest U.S.-based local-telephone company, with more than 20.2 million access lines, and the second largest cellular-mobile company. Its consolidated revenues and sales from continuing operations for 1991 totaled $19.6 billion with a consolidated net income of $1.8 billion. GTE's world headquarters is located in Stamford, Connecticut. The corporation employs approximately 160,000 people in various telephone and manufacturing operations in 48 states and 41 countries. GTE consists of the following operations:

GTE TELEPHONE OPERATIONS
GTE GOVERNMENT SYSTEMS
GTE MOBILE COMMUNICATIONS
GTE INFORMATION SERVICES
GTE SPACENET
GTE AIRFONE

Write a program to indent each GTE operation a specified number of spaces more than the previously displayed operation. The first line displayed is not to be indented. Example:

INPUT: Enter number of spaces: **2**

OUTPUT: **GTE TELEPHONE OPERATIONS**
GTE GOVERNMENT SYSTEMS
GTE MOBILE COMMUNICATIONS
GTE INFORMATION SERVICES
GTE SPACENET
GTE AIRFONE

1.5 GTE Data Services is a wholly owned subsidiary of GTE Corporation based in Stamford, Connecticut. GTEDS is one of the largest software development and information processing service companies in the United States, using 15 large mainframes to support approximately 125 major software systems. With corporate headquarters in Temple Terrace, Florida (next to Tampa), GTEDS is a part of GTE's Telephone Operations Group. More than 2,600 employees are located in the Tampa area, while approximately 5,000 are employed at four regional processing centers in the United States. The company was formed in October 1967 to provide information management and systems development services to customers nationwide. Since that time GTE Data Services has been providing low-cost, high-quality data processing, office automation and internal telecommunications product and services to GTE telephone operations in the United States, Canada and the Dominican Republic. Don A. Hayes was appointed president of GTE Data Services in July of 1988. Write a program to input a month and a year (of the form MM, YYYY) and display the number of WHOLE YEARS GTEDS has been operating. Examples:

INPUT: Enter M, Y: **5, 1992**

OUTPUT: **24 YEARS**

INPUT: Enter M, Y: **10, 1987**

OUTPUT: **20 YEARS**

1.6 Write a program to center a person's title and last name within a box of asterisks with dimensions 24 by 5. Input will be given on two separate lines, but both the title and the name must appear centered on the same line separated by one space. If one side must have one extra space, it must appear on the right side. Examples:

```
 INPUT: Enter title: COMMISSIONER
        Enter name: CASTOR

OUTPUT: ************************
        *                      *
        * COMMISSIONER CASTOR  *
        *                      *
        ************************

 INPUT: Enter title: PRESIDENT
        Enter name: HAYES

OUTPUT: ************************
        *                      *
        *   PRESIDENT HAYES    *
        *                      *
        ************************
```

1.7 GTE Data Services selects qualified candidates to participate in the Information Systems Orientation Program (ISOP). Selection is made from all individuals whose current position is at least a Salary Grade Level 5 and who have the potential and desire to occupy at least the positions of Systems Supervisor and Systems Manager. Write a program to accept the NAME of a selected candidate, their TITLE, and the GROUP that they represent, and then display the information in the form of a four-line statement as illustrated below. Examples:

```
 INPUT: Enter name: SCOTT
        Enter title: PROJECT LEADER
        Enter group: SERVICE ORDER DEVELOPMENT

OUTPUT: SCOTT IS A PROJECT LEADER WITHIN THE
        SERVICE ORDER DEVELOPMENT GROUP AND
        HAS BEEN SELECTED TO PARTICIPATE IN
        THE ISOP.


 INPUT: Enter name: RICK
        Enter title: PROJECT LEADER
        Enter group: CONVERSION

OUTPUT: RICK IS A PROJECT LEADER WITHIN THE
        CONVERSION GROUP AND
        HAS BEEN SELECTED TO PARTICIPATE IN
        THE ISOP.
```

1.8 One of the many benefits of working at GTEDS is the Personal Computer Purchase Program which allows employees to buy a personal computer through interest-free payroll deductions. At most $2000.00 may be borrowed. Write a program to display the dollar amount to be borrowed (preceded by a "$"), given the cost of the computer in dollars (without the dollar sign). Examples:

INPUT: Enter amount: **1255.70**
OUTPUT: **$1255.70**

INPUT: Enter amount: **2543.23**
OUTPUT: **$2000.00**

1.9 Debra, Lori, Sherry, Sue, and Tom all occupy the position of BIA (Business Information Analyst) in the SERVICE ORDER/TREATMENT Sub-application groups within CBSS at GTEDS. BIAs are knowledgeable in both the Telephone Operations business functions and the support of Data Processing systems. Write a program for the BIAs to display an acronym for a given set of business words separated by a space. The acronym is formed by concatenating the first letter of each word entered. Examples:

INPUT: Enter words: **BUSINESS INFORMATION ANALYST**
OUTPUT: **BIA**

INPUT: Enter words: **CUSTOMER BILLING SERVICES SYSTEM**
OUTPUT: **CBSS**

1.10 "Quality begins with U" is Scott's slogan for the SERVICE ORDER/TREATMENT SUPPORT group that he is supervising. His vision is to establish easy procedures for the technicians to follow enabling them to fit "quality" into their busy work schedule at GTEDS. One way in which Scott is implementing "quality" is by allowing all the technicians to meet together on a daily basis to enlighten one another about their critical day-to-day maintenance activities and how they responded to achieve a solution. These meetings help new technicians learn quicker and standardizes the way work is done by hearing how some of the more experienced programmers solve problems. Write a program to determine how many HOURS and MINUTES are devoted to implementing "quality" after 1 year (50 weeks) if N technicians meet 5 days a week for M minutes, given that N and M are input as positive integers. Examples:

INPUT: Enter number of technicians, N: **10**
Enter number of minutes, M: **15**

OUTPUT: **625 HOURS 0 MINUTES**

INPUT: Enter number of technicians, N: **7**
Enter number of minutes, M: **5**

OUTPUT: **145 HOURS 50 MINUTES**

2.1 Write a program to display several input lines of a speech in an indented outline format. The last line entered will be followed by a blank line. Each line that begins with a Roman numeral (I, II, III, IV) is displayed without any indentation. Each line that begins with a letter (A through H) is indented 4 spaces. Each line that begins with a number (1 through 9) is indented 8 spaces. Example:

```
 INPUT: Enter line: I. THE CROSS
        Enter line: A. HISTORY
        Enter line: 1. THE PERSIANS
        Enter line: 2. THE CARTHEAGANS
        Enter line: 3. THE ROMANS
        Enter line: B. MEDICAL PERSPECTIVE
        Enter line: C. HIS PERSONAL SUFFERING
        Enter line: II. THE LAST 7 STATEMENTS
        Enter line: III. 3 DAYS LATER
        Enter line: (press the return key)

OUTPUT: I. THE CROSS
            A. HISTORICAL PERSPECTIVE
                1. THE PERSIANS
                2. THE CARTHEAGANS
                3. THE ROMANS
            B. MEDICAL PERSPECTIVE
            C. HIS PERSONAL SUFFERING
        II. THE LAST 7 STATEMENTS
        III. 3 DAYS LATER
```

2.2 Bob is the project leader for the REPORTS SUPPORT group within CBSS. The Customer Billing Services System utilizes a utility called the REPORT FORMATTER GENERATOR (RFG) to create its reports. The RFG is a table driven utility which processes one to an infinite number of reports from a singe job-step. Write a program to display in words, the number of reports that the CBSS Director, Mick, is requesting. Input will be any integer between 1 and 99, inclusive. Examples:

```
 INPUT: Enter number: 25
OUTPUT: TWENTY-FIVE

 INPUT: Enter number: 91
OUTPUT: NINETY-ONE

 INPUT: Enter number: 16
OUTPUT: SIXTEEN

 INPUT: Enter number: 3
OUTPUT: THREE
```

2.3 GTE Data Services hires talented college graduates into their New Recruit Development Program. This is an intensive, innovative 14 week training program designed to develop individuals in the state-of-the-art technologies of GTE Data Services. Trainees will enter into the program as Programmers - New Recruit Associates and take part in lectures, lab session, and on-line training exercises in the areas of: COBOL, CICS, JCL, MVS/XA, VSAM Files, TSO/ISPF, C, Case Tools, DB2/SQL, Telecommunications, and Database Concepts. Recruiters generally hire those individuals with a BA/BS or higher degree in Computer Science, Math, Business, Engineering, Management Information Systems or Computer Information Systems. Other considerations are also reviewed as shown below. Write a program to accept as input the name of a potential recruit and his/her degree, and then display a menu of 7 other considerations as shown below. The computer then prompts the user to select up to 7 of the items listed (as a "string" in any order). Next, the computer is to clear the screen and display the name of the recruit followed by his/her degree followed by all the other considerations selected. Considerations are to be separated by a blank line and renumbered consecutively in the order they appear within the menu. Example:

INPUT: Enter name: **DOUG**
Enter degree: **BS IN COMPUTER SCIENCE**

OUTPUT: **1. DEMONSTRATED INTEREST IN INFORMATION MANAGEMENT.**
2. DEMONSTRATED LEADERSHIP SKILLS.
3. STRONG GPA/PERFORMANCE HISTORY.
4. AT LEAST TWO COURSES IN ANY PROGRAMMING LANGUAGE.
5. INTERNSHIP OR WORK EXPERIENCE.
6. EFFECTIVE ORAL AND WRITTEN COMMUNICATION SKILLS.
7. CAREER DEVELOPMENT POTENTIAL.

INPUT: Select up to 7 items: **17265**

OUTPUT: (screen is cleared)
DOUG
BS IN COMPUTER SCIENCE

1. DEMONSTRATED INTEREST IN INFORMATION MANAGEMENT.

2. DEMONSTRATED LEADERSHIP SKILLS.

3. INTERNSHIP OR WORK EXPERIENCE.

4. EFFECTIVE ORAL AND WRITTEN COMMUNICATION SKILLS.

5. CAREER DEVELOPMENT POTENTIAL.

2.4 Shelley has served as president of the Toastmasters Club at GTE Data Services. Each week the club meets to provide its members with a friendly atmosphere in which they can present effective speeches that inform, persuade, inspire, and entertain. Each meeting includes open evaluation, in which the speaker learns the audience's reaction to his or her presentation. One person is assigned the task of formally evaluating another person's speech. The speech is judged in regards to seven categories: speech value (interesting, meaningful), preparation (research, rehearsal), manner (direct, confident, sincere), organization (purposeful, clear), opening (attention-getting, led into topic), body of speech (logical flow; ideas supported by facts), and conclusion (effective, climatic). Write a program to accept an evaluator's verbal rating for the seven categories and to output the numerical value of each category, the average numerical rating (rounded to the nearest tenth), and the overall verbal rating, given the following rating scale:

1 = EXCELLENT
2 = ABOVE AVERAGE
3 = SATISFACTORY
4 = SHOULD IMPROVE
5 = MUST IMPROVE

The program then must display the equivalent overall verbal rating after rounding the displayed average numerical rating to the nearest whole number. Examples:

INPUT: Enter rating for speech value: **ABOVE AVERAGE**
Enter rating for preparation: **ABOVE AVERAGE**
Enter rating for manner: **SHOULD IMPROVE**
Enter rating for organization: **SATISFACTORY**
Enter rating for opening: **MUST IMPROVE**
Enter rating for body of speech: **ABOVE AVERAGE**
Enter rating for conclusion: **EXCELLENT**

OUTPUT: **SPEECH VALUE: 2**
PREPARATION: 2
MANNER: 4
ORGANIZATION: 3
OPENING: 5
BODY OF SPEECH: 2
CONCLUSION: 1

AVERAGE NUMERICAL RATING = 2.7
SPEECH RATING = SATISFACTORY

2.5 Write a program to display GTEDS MISSION statement formatted with at most N characters per line, where N is input as a number between 20 and 40 inclusive. Words in the following statement end with a space, dash, or period: "Be the customer-oriented leader and provider-of-choice of quality information products and services in the telecommunications marketplace and selected other related markets in support of GTE'S TELOPS goals." The last word displayed on a line may end with a dash. Example:

INPUT: Enter N: **28**

OUTPUT: **BE THE CUSTOMER-ORIENTED
LEADER AND PROVIDER-OF-
CHOICE OF QUALITY
INFORMATION PRODUCTS AND
SERVICES IN THE
TELECOMMUNICATIONS
MARKETPLACE AND SELECTED
OTHER RELATED MARKETS IN
SUPPORT OF GTE'S TELOPS
GOALS.**

2.6 Write a program to enter a paragraph with no commas, and change all periods to question marks if the sentence begins with WHAT, WHY, HOW, WHO, or WHERE. All sentences end with one of three characters: period (.), question mark (?), or exclamation point (!). Note: Input will be at most 125 characters; Output does not need to be formatted and may wrap around the end of the screen to the next line. Example:

INPUT: Enter paragraph: **WHAT IS TODAY'S DATE. MAY 2ND. WHERE ARE WE. IN TAMPA. WHY ARE WE HERE! WHOM ARE YOU. IS THIS FUN OR WHAT.**

OUTPUT: **WHAT IS TODAY'S DATE? MAY 2ND. WHERE ARE WE? IN TAMPA. WHY ARE WE HERE! WHOM ARE YOU. IS THIS FUN OR WHAT.**

2.7 Employees at GTEDS usually spend their time either in development work or in support work. Development builds a new system or adds new features to an existing system. Support, or maintenance, fixes functionality that currently is producing undesired results. Dan is the project leader for the SERVICE ORDER/TREATMENT SUPPORT sub-application group within CBSS at GTEDS. In order to respond promptly to the customers' needs, a beeper is carried by one person in the support group at all times. The primary beeper person is responsible for providing support to the customer and is the customer's primary contact for emergency situations. Write a program to display an alphabetical listing of all the employees that are in the office at a given time when the beeper "goes off". If no one is in the office at the given time, then display: NONE. Since the employees are on "flex time", their schedules may differ among themselves but will be consistent on a day to day basis. The following is a list of support personnel and their core hours (24-hour clock time), Monday through Friday unless otherwise indicated by a day off:

David	0700 - 1600	
Don	0800 - 1700	
Doug	0730 - 1630	
Grandville	1230 - 2100	
James	1130 - 2200	off on Monday
Jim	0900 - 1800	
John	0700 - 1600	
Linda	1230 - 2300	off on Friday
Marie	0700 - 1600	
Matt	1230 - 2300	off on Monday
Paula	0700 - 1600	
Robert	0800 - 1700	
Shelley	0630 - 1530	
Tom	1100 - 1930	

Examples:

INPUT: Enter time: **0730**
Enter day: **FRIDAY**

OUTPUT: **DAVID, DOUG, JOHN, MARIE, PAULA, SHELLEY**

INPUT: Enter time: **1930**
Enter day: **MONDAY**

OUTPUT: **GRANDVILLE, LINDA, TOM**

INPUT: Enter time: **1230**
Enter day: **SUNDAY**

OUTPUT: **NONE**

INPUT: Enter time: **0650**
Enter day: **THURSDAY**

OUTPUT: **SHELLEY**

2.8 Anita is the Supervisor of the TREATMENT DEVELOPMENT sub-application within the Customer Billing Services System (CBSS) at GTEDS. To insure the development of a high-quality product, Anita asks her project leader, Val, to conduct a formal inspection of their work, in accordance with company procedures. A formal inspection is a "rigorous structured review of a product to find defects." Val has designated that five of her co-workers will function as inspectors. Each inspector is assigned one of five titles and performs its function accordingly: moderator, reader, recorder, author, and inspector. The MODERATOR leads and guides the inspection. The READER sets the pace and paraphrases the material. The RECORDER lists the defects as they are detected. The AUTHOR provides clarification and explanation of the material when requested. Although everyone participates as an inspector, one person has the title of INSPECTOR. The seven co-workers that Val can choose from are Darlene, Jeff, Liz, Lori, Mary, Ping, and Will. Only Darlene and Will may function as a moderator. If Darlene or Will is the author, then the other person serves as the moderator; otherwise, there is a 50% chance that Val will select either person to serve as the moderator. The positions of reader, recorder, and inspector are chosen randomly (with equal probability) from among the remaining five co-workers. Write a program to simulate Val's selections for her inspection team given that one of the seven co-workers is input as the author. The following are POSSIBLE examples:

INPUT: Enter author's name: **WILL**

OUTPUT: **AUTHOR - WILL**
MODERATOR - DARLENE
READER - LIZ
RECORDER - JEFF
INSPECTOR - LORI

INPUT: Enter author's name: **WILL**

OUTPUT: **AUTHOR - WILL**
MODERATOR - DARLENE
READER - JEFF
RECORDER - LIZ
INSPECTOR - PING

INPUT: Enter author's name: **MARY**

OUTPUT: **AUTHOR - MARY**
MODERATOR - WILL
READER - JEFF
RECORDER - DARLENE
INSPECTOR - PING

2.9 The Customer Billing Services System (CBSS) was developed at GTE Data Services during a five year period from 1986 to 1990. With the help and commitment of GTE Telephone Operations, this tremendously complex billing system, consisting of approximately 5 million lines of code, was first installed in the summer of 1990 in Bethel, Pennsylvania. GTE was the first company in the telephony industry to introduce a new billing system that provides the extensive bill format and pricing flexibility. GTEDS is in the process of converting from the previous billing system, CRB, and the TOLL message system. These two large systems are converted to CBSS by the CONVERSION SYSTEM group under Bonnie's supervision. This group supports the conversion process to ensure the quality of data movement from the old systems to the new CBSS. The CONVERSION SYSTEM group, has been responsible for converting customer telephone numbers when an area code splits due to various reasons, such as an increase in population. Some telephone numbers remain the same while other telephone numbers take on a new area code. Write a program to enter two 3-digit area code numbers from a split, the number of names to be entered, and a list of names in the CONVERSION SYSTEM group that will be assigned to one of these two areas. After sorting the list of names alphabetically, the program will fictitiously assign the first half of the names to the smaller area code number and the last half to the larger area code number. If there is an odd number of people in the list, then the person in the middle of the list is assigned the smaller area code number. Examples:

```
 INPUT: Enter two area codes: 813, 811
        Enter number of names: 4
        Enter name: JENNIFER
        Enter name: JACKIE
        Enter name: BYRON
        Enter name: ESTHER

OUTPUT: 811 - BYRON
        811 - ESTHER
        813 - JACKIE
        813 - JENNIFER


 INPUT: Enter two area codes: 305, 307
        Enter number of names: 5
        Enter name: MARCELLE
        Enter name: MIKE
        Enter name: THERESA
        Enter name: CHARLOTTE
        Enter name: RICK

OUTPUT: 305 - CHARLOTTE
        305 - MARCELLE
        305 - MIKE
        307 - RICK
        307 - THERESA
```

2.10 Approximately 250 employees play in the GTE Data Services Golf League. Each week two teams of two golfers plays nine-holes against each other on one of four courses in the Tampa area. Tom, a supervisor for the REPORTS DEVELOPMENT sub-application in CBSS, is on the Rules Committee. The committee has determined that each player's handicap shall be calculated using USGA rules. A player's gross score is adjusted for handicap purposes by not counting any score over a triple bogey, (3 strokes over par). A player's current handicap determines the number of double or triple bogeys allowed: If the handicap is 9 or less, then the handicap indicates the number of scores that are limited to a double bogey (2 strokes over par), while the rest of the scores are limited to a bogey (1 stroke over par); otherwise, the scores are limited to a triple bogey for every handicap score over 9, and double bogeys for the rest of the scores. A handicap of 4 allows four double bogeys and 5 single bogeys. A handicap of 16 allows 7 triple bogeys and 2 double bogeys. The higher bogey limits are used up as they are needed from hole to hole. The handicap for the round is calculated by subtracting the course rating (par total in this case) from the adjusted total score. Write a program to display the adjusted score, the round handicap, and other statistics as shown below, given a golfer's current handicap (1 through 18) and gross scores for nine-holes of golf. Use the following pars for the nine holes:

Hole #:	1	2	3	4	5	6	7	8	9
Par:	5	4	4	4	3	4	4	3	5

Examples:

```
     INPUT: Enter handicap: 11
            Enter gross scores: 6,7,4,10,4,5,7,6,4

    OUTPUT: HOLE #:   1   2   3   4   5   6   7   8   9
            PAR:      5   4   4   4   3   4   4   3   5
            GROSS:    6   7   4  10   4   5   7   6   4
            ADJUST:   6   7   4   7   4   5   6   5   4

            PAR TOTAL: 36
            GROSS TOTAL: 53
            ADJUST TOTAL: 48
            ROUND HANDICAP: 12


     INPUT: Enter handicap: 5
            Enter gross scores: 7,4,7,6,7,8,7,6,7

    OUTPUT: HOLE #:   1   2   3   4   5   6   7   8   9
            PAR:      5   4   4   4   3   4   4   3   5
            GROSS:    7   4   7   6   7   8   7   6   7
            ADJUST:   7   4   6   6   5   6   5   4   6

            PAR TOTAL: 36
            GROSS TOTAL: 59
            ADJUST TOTAL: 49
            ROUND HANDICAP: 13
```

3.1 Write a program to move a triangle, made up of the acronym GTEDS, around the screen using the keys I, J, K, and M to move the triangle up, left, right, and down respectively. The triangle must initially appear in the approximate center of the screen. Once a valid directional key is pressed the triangle continuously shifts one column (or row) in the designated direction until either: 1) another valid directional key is pressed, causing the triangle to shift in another direction, or 2) the triangle's edge is about to go past the perimeter of the screen, in which case the triangle is to remain stationary until another directional key is pressed to send it away from (or along) the perimeter. The following triangle is to be displayed:

```
   G
  T T
 E   E
D     D
SDETGTEDS
```

3.2 Patricia is a wonderful manager over 5 sub-groups at CBSS. Each year she coordinates a Christmas party for her department that livens up the season. This past year each person bought and wrapped a small gift to play in the "Chinese gift exchange". After Derril picked a gift, his desirable Far-Side calendar gift was taken from him. Having been a biologist, he is knowledgeable of animal behavior and human behavior and able to enjoy the humor in interchanging animals with human roles. Fortunately, Santa Claus had placed another 1992 Far-Side calendar in his stocking. His calendar contains 314 pages of hilarious animal illustrations, 1 weekday of the year on each page with Saturdays and Sundays being combined on one page. If Derril has read X hilarious pages and torn them from his calendar, write a program to determine what day(s) of the year now appear(s) on his desk calendar. January 1, 1992 was a Wednesday. The program must display the day of the week, the month, and the day. Examples:

INPUT: Enter X: **5**

OUTPUT: **TUESDAY JANUARY 7**

INPUT: Enter X: **100**

OUTPUT: **MONDAY APRIL 27**

INPUT: Enter X: **309**

OUTPUT: **SATURDAY DECEMBER 26**
SUNDAY DECEMBER 27

INPUT: Enter X: **51**

OUTPUT: **SATURDAY FEBRUARY 29**
SUNDAY MARCH 1

3.3 Russ is the Supervisor of the SERVICE ORDER DEVELOPMENT sub-application within CBSS at GTEDS. His team uses the Programmer Work Station (PWS) to manage and coordinate enhancements made to their programs. PWS allows team members to communicate among themselves about their changes to the code for a particular module. After enhancements have been made in the TEST library environment of PWS, a request is made to release the changes to the BASE library environment, which will eventually be used in production. Write a program to accept the name of a person on the Service Order Development team, the program being enhanced, a flag (Y/N) indicating that changes are complete, and a flag (Y/N) to request release. If release is requested (Y), the computer will automatically mark the person's program as "completed", even if "N" was entered for this field. Release to base can only occur if everyone thus far has finished their enhancements on that program and at least one person has requested release. When a module is properly released, display the message MODULE XXXX HAS BEEN RELEASED, where XXXX is the four character program name. The program ends when all modules input have been released. Each module will be RELEASED only once. Examples:

```
     INPUT: Enter name, program: MIKE,TU03
            Enter completed, release: N,N

     INPUT: Enter name, program: PATRICK,TU03
            Enter completed, release: Y,Y

     INPUT: Enter name, program: MIKE,TU03
            Enter completed, release: Y,N

    OUTPUT: MODULE TU03 HAS BEEN RELEASED

 -------------------------------------------------
     INPUT: Enter name, program: LARRY,TU01
            Enter completed, release: Y,N

     INPUT: Enter name, program: DERRIL,TT00
            Enter completed, release: Y,N

     INPUT: Enter name, program: DERRIL,TU01
            Enter completed, release: Y,Y

    OUTPUT: MODULE TU01 HAS BEEN RELEASED

     INPUT: Enter name, program: DOUG,TT00
            Enter completed, release: N,N

     INPUT: Enter name, program: LARRY,TT00
            Enter completed, release: Y,Y

     INPUT: Enter name, program: DOUG,TT00
            Enter completed, release: Y,N

    OUTPUT: MODULE TT00 HAS BEEN RELEASED
```

3.4 A company would like to use an acronym as a phone number that is easy for the public to remember. They would like a word that can be used for the last several digits of their number, where each letter corresponds to a particular digit. Each possible word will be 4 or 5 letters long. Write a program to display all possible acronym phone numbers (alphabetically by the acronyms used) for an input number of the format XXX-XXXX. Assume that at least one word will satisfy the requirements given the following word list options:

```
AGENT  SOAP   MONEY  JEWEL  BALL   LOANS  CARE   SAVE   CALL
PAVE   KEEP   KINGS  KNIFE  KNOCK  JOINT  JUICE  LOBBY  RATE
```

Use the following letters to correspond to the digits 2 to 9:

```
A B C   D E F   G H I   J K L   M N O   P R S   T U V   W X Y
  2       3       4       5       6       7       8       9
```

Examples:

INPUT: Enter phone #: **555-3935**

OUTPUT: **55J-EWEL**

INPUT: Enter phone #: **555-2255**

OUTPUT: **555-BALL**
555-CALL

3.5 Write a program to display seven 7-digit squares which contain no duplicate digits in the octonary (base 8) system. One such number is 1567204 because it is 1242 * 1242 in base 8. Output each 7-digit number followed by 2 spaces and its square root. Display each successive entry on a separate line in ascending order. The following example illustrates the format of the output, but it only gives the first solution. Example:

```
OUTPUT: 1567204  1242
        #######  ####
        #######  ####
        #######  ####
        #######  ####
        #######  ####
        #######  ####
```

3.6 GTE Data Services has a highly talented group of classroom instructors that equip qualified New Recruits with a wide range of skill sets necessary to succeed in the GTEDS environment. The trainers in the program impart to their students not only technical skills, but personal insights into corporate life and the needed people-skills to succeed in the company. In addition to teaching the Recruits a "right" way to solve programming problems, the instructors also lead them on a journey to discover a solution that is highly "efficient." Write a program that is "efficient", given the following information:

Every integer, N, greater than 17 can be written as the sum of three distinct integers, each greater than 1, such as:

X + Y + Z = N

where X, Y, and Z are pairwise relatively prime with respect to each other. When N is equal to 20, there are six sets of integers that satisfy the equation:

2+5+13, 2+7+11, 3+4+13, 3+7+10, 4+5+11, 4+7+9.

Notice that although 3+6+11 equals 20, the set does not satisfy the conditions since the numbers 3 and 6 have a common factor: 3.

Write an "efficient" program to generate the smallest combination of X, Y, and Z (with X as small as possible, then Y, then Z) that satisfies the equation for an input number, N, between 18 and 32700. Display the solution in the following format:

X + Y + Z = N

with X < Y < Z. Examples:

INPUT: Enter N: **20**
OUTPUT: **2 + 5 + 13 = 20**

INPUT: Enter N: **184**
OUTPUT: **2 + 3 + 179 = 184**

INPUT: Enter N: **185**
OUTPUT: 3 + 19 + 163 = 185

INPUT: Enter N: **32693**
OUTPUT: **3 + 13 + 32677 = 32693**

INPUT: Enter N: **32694**
OUTPUT: **2 + 3 + 32689 = 32694**

3.7 Doug and Dan have each served as the captain of the New Recruits indoor soccer team. Four other GTEDS New Recruit employees have consistently play on the team: Andy, Jack, Mike, and Yehia. No more than six players may play on the field at any time. Substitutions may be made for any player at any time. Write a program to accept, as input, 1 to 3 extra players that come for the game, and then display all possible combinations of six players (listed alphabetically) that can play together. Display each list numbered and ordered alphabetically among the lists, assuming that the names are concatenated within the list. If the number of lines of lists exceeds the number of lines on the screen, then pause each screen of output, allowing a key to be pressed to display then next screen of output. Examples:

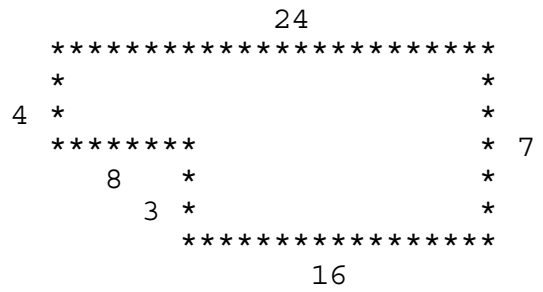
```
 INPUT: Enter number of substitutes: 1
        Enter name: TONY

OUTPUT: 1 ANDY,DAN,DOUG,JACK,MIKE,TONY
        2 ANDY,DAN,DOUG,JACK,MIKE,YEHIA
        3 ANDY,DAN,DOUG,JACK,TONY,YEHIA
        4 ANDY,DAN,DOUG,MIKE,TONY,YEHIA
        5 ANDY,DAN,JACK,MIKE,TONY,YEHIA
        6 ANDY,DOUG,JACK,MIKE,TONY,YEHIA
        7 DAN,DOUG,JACK,MIKE,TONY,YEHIA


 INPUT: Enter number of substitutes: 3
        Enter name: PAUL
        Enter name: DEAN
        Enter name: ROB

OUTPUT: 1 ANDY,DAN,DEAN,DOUG,JACK,MIKE
        2 ANDY,DAN,DEAN,DOUG,JACK,PAUL
        3 ANDY,DAN,DEAN,DOUG,JACK,ROB
        4 ANDY,DAN,DEAN,DOUG,JACK,YEHIA
        5 ANDY,DAN,DEAN,DOUG,MIKE,PAUL
        6 ANDY,DAN,DEAN,DOUG,MIKE,ROB
        :
        :
       63 DAN,DEAN,DOUG,JACK,ROB,YEHIA
       64 DAN,DEAN,DOUG,MIKE,PAUL,ROB
        :
        :
       81 DEAN,DOUG,JACK,PAUL,ROB,YEHIA
       82 DEAN,DOUG,MIKE,PAUL,ROB,YEHIA
       83 DEAN,JACK,MIKE,PAUL,ROB,YEHIA
       84 DOUG,JACK,MIKE,PAUL,ROB,YEHIA
```

Note: lines 7 through 62 and lines 65 through 80 are not displayed in the example, but must appear within the actual output, allowing a key to be pressed after a screen full of lines has been displayed.

3.8 The Customer Billing Services System (CBSS) at GTEDS prints its customers' phone bills at different times to suit the customers' needs. Bills are produced on one of 10 cycles, starting on the first of the month and occurring every 3 days afterwards: 1, 4, 7, 10, 13, 16, 19, 22, 25, 28. The Bill Due Date is assigned to an account by adding a specified number of days to the account's Bill Date (the day the bill was produced). In many cases, if the due date falls on a Saturday, Sunday, or holiday, the system moves the due date to the next weekday that is not a holiday. Write a program to display the BILL DATE and the DUE DATE given the following as input: the month of the bill in 1992, the cycle number, the number of days (less than 25) to add to the Bill Date, and a set of holidays in the form MM, DD (terminated by 0, 0). January 1, 1992 was a Wednesday. Examples:

```
 INPUT: Enter month of bill: 1
        Enter cycle number: 6
        Enter number of days: 15
        Enter holiday MM, DD: 1, 1
        Enter holiday MM, DD: 1, 31
        Enter holiday MM, DD: 0, 0

OUTPUT: BILL DATE: THURSDAY JANUARY 16
        DUE DATE: MONDAY FEBRUARY 3


 INPUT: Enter month of bill: 5
        Enter cycle number: 1
        Enter number of days: 24
        Enter holiday MM, DD: 5, 25
        Enter holiday MM, DD: 0, 0

OUTPUT: BILL DATE: FRIDAY MAY 1
        DUE DATE: TUESDAY MAY 26
```

3.9 GTE Data Services is located in four large information processing centers in Temple Terrace, Florida; San Angelo, Texas; Sacramento, California; and Fort Wayne, Indiana. GTEDS employs over 5,000 data processing, software, and network professionals within the buildings located in the four areas. The buildings have many different rooms and cubicles. Write a program to calculate the area of a room in the shape of a polygon with perpendicular corners, given a series of movements describing its shape. After the program accepts the number of vertical and horizontal sides in the room, it then accepts a list of successive direction-distance pairs, starting from an arbitrary corner. Directions will be U, D, R, and L to indicate Up, Down, Right, and Left respectively. Each direction will be followed by a distance in feet, less than 25. Each room described will have at most 10 corners and will have both a length and a width less than 25 feet. The first example uses a polygon room with the shape and dimensions of:

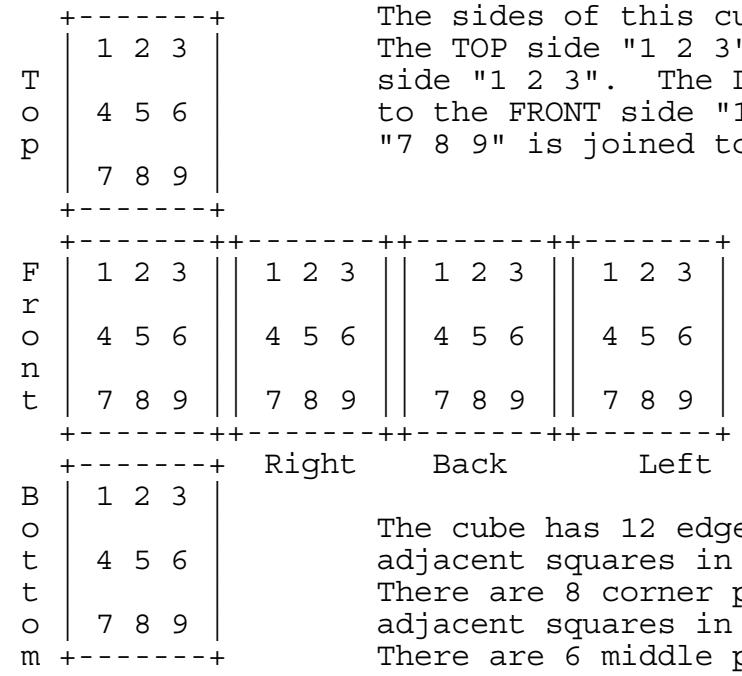


Examples:

```
 INPUT: Enter number of sides: 6
        Enter movement: U3
        Enter movement: L8
        Enter movement: U4
        Enter movement: R24
        Enter movement: D7
        Enter movement: L16

OUTPUT: AREA = 144 SQUARE FEET


 INPUT: Enter number of sides: 10
        Enter movement: R8
        Enter movement: U2
        Enter movement: R6
        Enter movement: D10
        Enter movement: L10
        Enter movement: U3
        Enter movement: L9
        Enter movement: U7
        Enter movement: R5
        Enter movement: D2

OUTPUT: AREA = 147 SQUARE FEET
```

3.10 The Rubik's Cube, invented by the Hungarian Erno Rubik in 1975, is a three by three by three inch cube. A cube has six sides or faces which contain a different color. Each face is divided up into nine squares--a total of 54 squares on the cube. A brilliant inner mechanism of a spring loaded spindle allows sides of the cube to be rotated either vertically or horizontally and independent of the other sides. In the cube's pristine condition it has a solid color on each face, but is scrambled by just a few random moves. The cube comes in six different colors, of which, the original Rubik's Cube contains: white, yellow, orange, red, green, and blue. The object is to get the cube back to its original positions so that all six sides have nine identical squares with respect to their colors. If the cube is taken apart and randomly reassembled, then there is only a 1 in 12 chance that the puzzle can be solved. An almost sure way to make the cube impossible to solve is to remove the 54 square colors and randomly place them back on the squares of the cube.

Write a program to accept as input, 6 sides of 9 square color symbols for the Rubik's Cube, and to then determine the reasons for judging the cube as insolvable, as given on the next page. One letter color symbols (W, Y, O, R, G, B) will be entered from left to right, top to bottom, for the following sides (in order): top, front, right, back, left, bottom. The order of input is illustrated below by numbers for each side of the cube:



The sides of this cube are connected as shown. The TOP side "1 2 3" is joined to the BACK side "1 2 3". The LEFT side "3 6 9" is joined to the FRONT side "1 4 7". The BOTTOM side "7 8 9" is joined to the BACK side "7 8 9".

The cube has 12 edge pieces made up of 2 adjacent squares in positions 2,4,6,8. There are 8 corner pieces made up of 3 adjacent squares in positions 1,3,7,9. There are 6 middle pieces in position 5.

*** CONTINUED ON NEXT PAGE ***

After the program accepts the color symbols on the six sides, then display one or two of the following statements (in order), with the first statement displayed conditionally:

If two of the middle squares have the same color, display:

COLORS ON MIDDLE SQUARES ARE NOT UNIQUE

Display the following message on the next (or first) line:

NUMBER OF EDGE PIECES HAVING SAME COLOR: ##

where ## is a number between 0 and 12.

Examples:

```
INPUT: Enter colors on top:    R,G,R,O,W,B,W,B,Y
       Enter colors on front:  G,O,O,R,Y,G,B,W,B
       Enter colors on right:  W,G,Y,W,O,B,Y,O,R
       Enter colors on back:   W,G,B,Y,R,Y,G,R,G
       Enter colors on left:   Y,O,G,W,R,B,B,O,R
       Enter colors on bottom: G,W,O,Y,B,R,O,Y,W

OUTPUT: COLORS ON MIDDLE SQUARES ARE NOT UNIQUE
        NUMBER OF EDGE PIECES HAVING SAME COLOR: 3
```

Note: Edge "Back 2" and Edge "Top 2" have same color: G
Edge "Front 4" and Edge "Left 2" have same color: O
Edge "Front 8" and Edge "Bottom 2" have same color: W

```
INPUT: Enter colors on top:    B,R,G,B,W,G,Y,B,R
        Enter colors on front: W,G,B,W,O,Y,G,R,R
       Enter colors on right:  W,R,W,Y,G,O,Y,W,B
       Enter colors on back:   O,B,R,G,R,O,O,W,B
       Enter colors on left:   G,O,O,Y,Y,G,Y,O,R
       Enter colors on bottom: W,Y,B,R,B,Y,O,W,G

OUTPUT: NUMBER OF EDGE PIECES HAVING SAME COLOR: 2
```

Note: Edge "Front 6" and Edge "Right 4" have same color: Y
Edge "Bottom 8" and Edge "Back 8" have same color: W

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '93

1.1 Write a program to display the following six lines consisting of the acronym for GTE Data Services Inc.:

```
GTEDS GTEDS GTEDS GTEDS GTEDS GTEDS
GTEDS  GTEDS  GTEDS  GTEDS  GTEDS
GTEDS   GTEDS   GTEDS   GTEDS
GTEDS    GTEDS    GTEDS
GTEDS     GTEDS
GTEDS
```

1.2 Every couple of months, GTE Data Services hires qualified individuals for their New Recruit Development program. This program features 14 and 15 week curriculums designed to train programmers in the standards and procedures of GTE Data Services. The course is "pass/fail" and successful students are placed in programmer positions in the company with a competitive starting salary. Entrance into the program is very competitive since GTEDS pools from as many as 300 candidates each period to select approximately 15 individuals by reviewing resumes, conducting telephone interviews, administering aptitude tests, and holding panel interviews.

Write a program to determine how many PROGRAMMERS have been placed in the company if N classes, starting with 15 students, have graduated with a total of M students dropping from the course. Examples:

INPUT: Enter N: **19**	INPUT: Enter N: **10**
Enter M: **9**	Enter M: **5**
OUTPUT: **276 PROGRAMMERS**	OUTPUT: **145 PROGRAMMERS**

1.3 GTE is the largest U.S.-based local-telephone company with domestic and international operations serving more than 20.7 million access lines in 40 states, Canada, and Latin America.

Write a program to display the formatted numeric figure of N million access lines, where N is input as a real number greater than 1 and less than 1000 and contains at most 4 digits. Examples:

INPUT: Enter N: **20.7**

OUTPUT: **20,700,000 ACCESS LINES**

INPUT: Enter N: **3.456**

OUTPUT: **3,456,000 ACCESS LINES**

1.4 The University of South Florida (USF) is ranked as the second largest university in the Southeast, having had a fall 1992 enrollment of over 34,000 students. Approximately half of these students are full-time and the other half are part-time. Approximately 57% of the student population is female. USF is located on five campuses: Tampa, St. Petersburg, Fort Myers, Lakeland, and Sarasota.

Write a program to accept as input the number of students attending each of the campuses and display the total number of STUDENTS enrolled at USF. Example:

```
 INPUT: Enter # at Tampa: 27319
        Enter # at St. Petersburg: 3010
        Enter # at Fort Myers: 1329
        Enter # at Lakeland: 856
        Enter # at Sarasota: 1846

OUTPUT: 34360 STUDENTS
```

1.5 GTE Data Services selects qualified candidates to participate in the Information Systems Orientation Program (ISOP). Selection is made from all individuals whose current position is at least a Salary Grade Level 5 and who have the potential and desire to occupy at least the positions of Systems Supervisor and Systems Manager.

Write a program to accept the NAME of a selected candidate, their Salary Grade LEVEL, and whether they DESIRE to occupy at least the positions of Systems Supervisor and Systems Manager (YES or NO). Display a statement indicating whether or not the person entered is A POSSIBLE CANDIDATE FOR ISOP. Examples:

```
 INPUT: Enter name: SCOTT
        Enter level: 6
        Enter desire: YES

OUTPUT: SCOTT IS A POSSIBLE CANDIDATE FOR ISOP


 INPUT: Enter name: DOUG
        Enter level: 3
        Enter desire: YES

OUTPUT: DOUG IS NOT A POSSIBLE CANDIDATE FOR ISOP


 INPUT: Enter name: JOE
        Enter level: 7
        Enter desire: NO

OUTPUT: JOE IS NOT A POSSIBLE CANDIDATE FOR ISOP
```

1.6 The New Recruit Development program features either a C/UNIX or MVS/COBOL curriculum based on the business needs of GTE Data Services. Some of the selection criteria used to hire employees for this program include:

- BA/BS in Computer Science, Math, Business, Engineering, MIS or CIS preferred
- Courses in any programming language
- Demonstrated interest in and commitment to information Management
- Experience in mainframe, mini or microcomputers
- Demonstrated leadership skills
- Strong GPA/Performance history
- Computer-related work experience
- Effective oral and written communication skills
- Career development potential
- Team player

If GTEDS is hiring for the C/UNIX curriculum then the following is a list of additional preferred skills: C, UNIX, ANSI SQL, OSF/MOTIF, SHELL PROGRAMMING.

If GTEDS is hiring for the MVS/COBOL curriculum, then the following is a list of additional preferred skills: COBOL, JCL, MVS/ESA, TSO/ISPF, VSAM, ANSI SQL, DB2, IMS.

Write a program to enter the curriculum for which the recruiter is hiring and display all the preferred skills for that curriculum. Examples:

```
 INPUT: Enter curriculum: MVS/COBOL

OUTPUT: COBOL
        JCL
        MVS/ESA
        TSO/ISPF
        VSAM
        ANSI SQL
        DB2
        IMS


 INPUT: Enter curriculum: C/UNIX

OUTPUT: C
        UNIX
        ANSI SQL
        OSF/MOTIF
        SHELL PROGRAMMING
```

1.7 Write a program to display the first N letters of the alphabet, where N is input as a number less than or equal to 26. Examples:

INPUT: Enter N: **6**

OUTPUT: **ABCDEF**

INPUT: Enter N: **25**

OUTPUT: **ABCDEFGHIJKLMNOPQRSTUVWXY**

1.8 Although money is not the primary reason why people remain computer programmers, an increase in salary does provide extra motivation. An employee's salary usually increases in proportion to his/her performance on the job. Write a program to accept a computer programmer's salary (greater than 20,000 and less than 90,000) and his/her "performance rating" and then display the new salary to be received (rounded to the nearest penny) based on the percentage rates listed below:

Performance	Increase
EXCELLENT	10%
ABOVE AVERAGE	7%
GOOD	5%

Examples:

INPUT: Enter salary: **25000.50**
Enter rating: **EXCELLENT**

OUTPUT: **NEW SALARY = $27500.55**

INPUT: Enter salary: **50000**
Enter rating: **ABOVE AVERAGE**

OUTPUT: **NEW SALARY = $53500.00**

INPUT: Enter salary: **30500.19**
Enter rating: **GOOD**

OUTPUT: **NEW SALARY = $32025.20**

1.9 When a GTE customer requests either telephone service, modifications to an existing service, or disconnection of service, the telephone company enters a "Service Order" into their computer system called SORCES (Service Office Record and Computer Entry System). This service order is then sent to the Customer Billing Services System (CBSS), which is an advanced and highly flexible customer billing system that produces detailed bills that are easy to read and understand.

Under Mike's direction, Russ and Scott supervise the development and maintenance of the Service Order sub-application programs within CBSS. Six of the most commonly used types of orders that these programs receive from SORCES are listed below:

I = INSTALL - an initial order to establish service for a new customer.

C = CHANGE - an order to make a change to existing service, such as adding a pricing plan, or changing a phone number, or having the desk phone removed and a wall phone added.

R = RECORDS - an order to create a non-service affecting change to the customer's account. Examples include: name changes, address corrections, and changing the day when a customer receives his bill.

O = OUT - an order issued to disconnect service.

F = FROM - an order to disconnect service at an old address (similar to an 'O' order) and is the first half of an F & T combination.

T = TO - an order to install service at a new location (similar to an 'I' order) and is the second half of an F & T combination.

Write a program to display the Service Order abbreviation if given the descriptive order, and to display the descriptive order if given the abbreviation for an order. Examples:

```
  INPUT: Enter order: INSTALL
OUTPUT: I

 INPUT: Enter order: F
OUTPUT: FROM

 INPUT: Enter order: I
OUTPUT: INSTALL

 INPUT: Enter order: CHANGE
OUTPUT: C
```

1.10 The average high school grade point average of the freshman class at the University of South Florida is 3.24. At USF the GPA is derived by dividing the total grade points by the total credits.

Write a program to enter 5 class grades, each having the same number of credits, and display the GPA for the semester. The values of the grades are as follows:

A = 4,
B = 3,
C = 2,
D = 1,
F = 0,
I = 0,
M = 0,
W is a withdrawal from the class.

A grade of 'I' (incomplete) and 'M' (missing) are counted as a grade of 'F' until resolved; however, a grade of 'W' (withdrawal) is not tallied since the class has been dropped (one less class of credits is used for dividing the points). The GPA must be displayed with 3 digits to the right of the decimal and rounded to the nearest 0.001. Examples:

INPUT: Enter grade: **A**
Enter grade: **C**
Enter grade: **F**
Enter grade: **I**
Enter grade: **B**

OUTPUT: **GPA = 1.800**

Note: In this example (4 + 2 + 0 + 0 + 3) / 5 = 1.800.

INPUT: Enter grade: **C**
Enter grade: **M**
Enter grade: **A**
Enter grade: **W**
Enter grade: **D**

OUTPUT: **GPA = 1.750**

Note: In this example (2 + 0 + 4 + 1) / 4 = 1.750.
The grade points are divided by 4 classes instead of 5 classes because 1 class was dropped, indicated by a grade of 'W'.

2.1 Write a program to randomly generate N numbers between X and Y inclusive, where N is input as an integer less than 11 while X and Y are input as integers between -99 and 99 inclusive. Display all numbers on one line with one space separating each number. Possible examples (outputs will vary):

```
 INPUT: Enter N: 5
        Enter X, Y: -15, -4

Possible OUTPUT: -13 -5 -15 -5 -8


 INPUT: Enter N: 10
        Enter X, Y: 55, -1

Possible OUTPUT: 23 34 0 16 55 2 6 53 17 28


 INPUT: Enter N: 5
        Enter X, Y: -4, -15

Possible OUTPUT: -6 -11 -7 -4 -12
```

2.2 Write a program to help produce an organizational work-chart for N people within a department at GTEDS. Input will consist of technician names, each followed by his/her grade level title. The names and titles are to be displayed in descending order by title and then ascending order by name. The grade level titles are as follows (from lowest to highest): P, PA, SA, SE, SSE, ASE, SASE.

```
 INPUT: Enter N: 6
        Enter name: PAULA
        Enter title: SA
        Enter name: MARION
        Enter title: ASE
        Enter name: KEVIN
        Enter title: P
        Enter name: DON
        Enter title: SE
        Enter name: DOUG
        Enter title: P
        Enter name: DERRIL
        Enter title: SE

OUTPUT: MARION - ASE
        DERRIL - SE
        DON - SE
        PAULA - SA
        DOUG - P
        KEVIN - P
```

2.3 Write a program to accept as input a COBOL declaration of a record and display the fields indented appropriately. Each field begins with a two digit level number. Position all fields beginning with level number 01 in the first column while each successive field is indented 4 spaces more than the previous field if the level number is greater than the previous level number; if the level number is less than the previous level number then the field is indented 4 spaces less than the previous field; and if the level number is the same as the previous level number then the field is lined up in the same column as the previous field. The first line input will begin with the level number 01, and the last line input will be a blank line. Examples:

```
 INPUT: Enter field: 01 WS-NAME  PIC X(15).
        Enter field: 01 WS-ADDRESS.
        Enter field: 05 WS-STREET PIC X(20).
        Enter field: 05 WS-CITY-ST.
        Enter field: 10 WS-CITY  PIC X(20).
        Enter field: 10 WS-STATE PIC X(02).
        Enter field: 07 WS-ZIP-9.
        Enter field: 15 WS-ZIP   PIC X(05).
        Enter field: 15 WS-ZIP-4 PIC X(04).
        Enter field: 10 FILLER PIC X(10).
        Enter field: 20 FILLER PIC X(15).
        Enter field: 01 WS-DATA PIC X(80).
        Enter field: (Press the Enter key)

OUTPUT: 01 WS-NAME  PIC X(15).
        01 WS-ADDRESS.
            05 WS-STREET PIC X(20).
            05 WS-CITY-ST.
                10 WS-CITY  PIC X(20).
                10 WS-STATE PIC X(02).
            07 WS-ZIP-9.
                15 WS-ZIP   PIC X(05).
                15 WS-ZIP-4 PIC X(04).
            10 FILLER PIC X(10).
                20 FILLER PIC X(15).
        01 WS-DATA PIC X(80).
```

2.4 Write a program to translate a word into a number by substituting each letter with its position in the alphabet, and then display the amount of blocks of contiguous even digits and odd digits within the number. Example:

```
 INPUT: Enter word: CONTEST

OUTPUT: NUMBER = 315142051920
        BLOCKS = 4


 INPUT: Enter word: WAY

OUTPUT: NUMBER = 23125
        BLOCKS = 4
```

Note: the first example has 4 blocks: 3151, 420, 519 and 20;
the second example has 4 blocks: 2, 31, 2, and 5.

2.5 Write a program to enter N telephone numbers and display them with dashes in the appropriate positions: after the NPA (the first three digits), and after the NXX (the next three digits). Display a blank line between numbers with the same NPA but a different NXX, and display 2 blank lines between numbers with a different NPA. Numbers will be input in ascending order. Display the total amount of telephone numbers within an NPA on the last line displayed within that section. Examples:

```
 INPUT: Enter N: 8
        Enter #: 1234567890
        Enter #: 1234568907
        Enter #: 1235678901
        Enter #: 1235679012
        Enter #: 1235679999
        Enter #: 2345678901
        Enter #: 3456789012
        Enter #: 3457890123

OUTPUT: 123-456-7890
        123-456-8907

        123-567-8901
        123-567-9012
        123-567-9999  TOTAL FOR NPA OF 123 = 5


        234-567-8901  TOTAL FOR NPA OF 234 = 1


        345-678-9012

        345-789-0123  TOTAL FOR NPA OF 345 = 2
```

2.6 Write a program to calculate the cost of purchasing several grocery store products (A - Z), where a person has one or more coupons for products (A - Z). The first part of the input will consist of a set of products with their prices (more than $1 and less than $10), ended by entering a fictitious product (9). The second part of the input will consist of a set of coupons for products with their discounts (less than $1), ended by entering a fictitious coupon (9). Output will be the TOTAL dollar amount of the bill (not including tax), after all applicable coupons have been used. If there is more than one coupon for the same product bought, then the largest coupon discount will be used. A coupon discount for a product may only be used once. Example:

```
     INPUT: Enter product: F
            Enter price: 1.50
            Enter product: B
            Enter price: 1.75
            Enter product: C
            Enter price: 5.00
            Enter product: B
            Enter price: 1.75
            Enter product: 9

            Enter coupon: B
            Enter discount: 0.50
            Enter coupon: D
            Enter discount: 0.60
            Enter coupon: C
            Enter discount: 0.75
            Enter coupon: B
            Enter discount: 0.65
            Enter coupon: B
            Enter discount: 0.70
            Enter coupon: 9

    OUTPUT: TOTAL = $7.90
```

Note: 7.90 = 1.50 + (1.75 - 0.70) + (5.00 - 0.75) + (1.75 - 0.65)

```
     INPUT: Enter product: C
            Enter price: 8.50
            Enter product: B
            Enter price: 7.75
            Enter product: 9

            Enter coupon: B
            Enter discount: 0.50
            Enter coupon: 9

    OUTPUT: TOTAL = $15.75
```

2.7 There are three standard formats for the DATE data type in a particular COBOL system:

YYYY-MM-DD	ISO (International Standards Organization)
MM-DD-YYYY	AMERICAN
DD-MM-YYYY	EUROPEAN

Write a program to accept as input a date in one of the three formats and display the date in the other two formats in the order given in the above list. Examples:

INPUT: Enter format: **AMERICAN**
Enter date: **02-04-1993**

OUTPUT: **ISO = 1993-02-04**
EUROPEAN = 04-02-1993

INPUT: Enter format: **ISO**
Enter date: **1994-12-31**

OUTPUT: **AMERICAN = 12-31-1994**
EUROPEAN = 31-12-1994

INPUT: Enter format: **EUROPEAN**
Enter date: **16-10-1993**

OUTPUT: **ISO = 1993-10-16**
AMERICAN = 10-16-1993

2.8 Write a program to reverse the order of words in one or two sentences input, where each word within a sentence is separated by one space and each sentence is ended by a period. Two spaces will separate the first and second sentences. Examples:

INPUT: Enter sentence: **I AM HAPPY. SO ARE YOU.**

OUTPUT: **HAPPY AM I. YOU ARE SO.**

INPUT: Enter sentence: **I AM ENJOYING THIS COMPUTER CONTEST.**

OUTPUT: **CONTEST COMPUTER THIS ENJOYING AM I.**

2.9 Write a program to enter a 4 x 4 matrix of positive integers less than 10 and display the 4 smallest elements and their location(s) within the matrix. If a smallest element occurs more than once, list the locations in ascending order by rows and then by columns, each separated by a comma. Example:

```
 INPUT: Enter row 1: 1, 3, 5, 7
        Enter row 2: 9, 8, 3, 1
        Enter row 3: 6, 7, 8, 9
        Enter row 4: 9, 1, 5, 5

OUTPUT: 1. SMALLEST = 1 OCCURS AT (1,1), (2,4), (4,2)
        2. SMALLEST = 3 OCCURS AT (1,2), (2,3)
        3. SMALLEST = 5 OCCURS AT (1,3), (4,3), (4,4)
        4. SMALLEST = 6 OCCURS AT (3,1)
```

2.10 GTE Data Services Inc. is a wholly owned subsidiary of GTE Corporation based in Stamford, Connecticut. GTEDS is one of the largest software development and information processing service companies in the United States, using 15 large mainframes to support approximately 125 major software systems. With corporate headquarters in Temple Terrace, Florida (next to Tampa), GTEDS is a part of GTE's Telephone Operations Group. More than 2,600 employees are located in the Tampa area, while approximately 5,000 are employed at GTEDS' four regional processing centers in the United States. **GTEDS was incorporated on Oct. 25, 1967** to provide information management and systems development services to customers nationwide. Since that time GTE Data Services has been providing low-cost, high-quality data processing, office automation and internal telecommunications product and services to GTE telephone operations in the United States, Canada and the Dominican Republic. Don A. Hayes was appointed president of GTE Data Services in July of 1988, and recently hosted GTEDS' 25th Anniversary Celebration at the University of South Florida (USF) Special Events Center.

Write a program to accept as input a month, day, and year (less than 2000), and then display the number of days GTEDS has been operating. Examples:

```
 INPUT: Enter month: 10          INPUT: Enter month: 2
        Enter day: 27                   Enter day: 4
        Enter year: 1967                Enter year: 1993

OUTPUT: 2 DAYS                  OUTPUT: 9234 DAYS
```

3.1 Write a program that will allow the user to move and position the cursor (a pound sign) on the screen by using appropriate keys (designated by the programmer). The program then allows the user to press 1, 2, 3, or 4, and displays a square made up of **GTEDS** relative to the cursor's position (as shown below) with the number centered in the square. If the square to be displayed will not fit on the screen, then display the error message **OFF THE SCREEN** on the top line of the screen; otherwise, display the appropriate square as shown below, relative to the cursor:

```
  #                   #
  G T E D S     G T E D S     G T E D S     G T E D S
  T       D     T       D     T       D     T       D
  E   1   E     E   2   E     E   3   E     E   4   E
  D       T     D       T     D       T     D       T
  S D E T G     S D E T G     S D E T G     S D E T G
                                        #     #
```

Examples:

INPUT: (**Move cursor** to top right corner and press **2**)

OUTPUT:
```
          #
  G T E D S
  T       D
  E   2   E
  D       T
  S D E T G
```

INPUT: (**Move cursor** to bottom right corner and press **3**)

OUTPUT:
```
  G T E D S
  T       D
  E   3   E
  D       T
  S D E T G
          #
```

INPUT: (**Move cursor** to top left corner and press **4**)

OUTPUT: **OFF THE SCREEN**
(Note: message must appear on the top line of screen)

INPUT: (**Move cursor** to bottom left corner and press **1**)

OUTPUT: **OFF THE SCREEN**
(Note: message must appear on the top line of screen)

3.2 Write a program to find the value of X that satisfies an input equation with one arithmetic symbol. All numbers in the equation will be positive integers less than 1000, and the arithmetic symbols that will be used are +, -, *, and /. The following examples illustrate the procedure for entering equations such as X+5=9, 7-6=X, 150=X*5, and 6=24/X. Examples:

```
 INPUT: Enter value: X
        Enter symbol: +
        Enter value: 5
        Enter symbol: =
        Enter value: 9

OUTPUT: X = 4
```

```
 INPUT: Enter value: 7
        Enter symbol: -
        Enter value: 6
        Enter symbol: =
        Enter value: X

OUTPUT: X = 1
```

```
 INPUT: Enter value: 150
        Enter symbol: =
        Enter value: X
        Enter symbol: *
        Enter value: 5

OUTPUT: X = 30
```

```
 INPUT: Enter value: 6
        Enter symbol: =
        Enter value: 24
        Enter symbol: /
        Enter value: X

OUTPUT: X = 4
```

3.3 Write a program to accept as input a string of several unique digits (1 through 7), and display all possible combinations of the digits which sum to an input number (**given in base 8**). Each set of single digit addends are to be separated by a plus sign and must appear in the same order as given in the input string. Each line of addends may be in any order. Examples:

```
 INPUT: Enter digits: 6527
        Enter sum: 15

OUTPUT: 6+5+2 = 15
        6+7 = 15
```

or

```
OUTPUT: 6+7 = 15
        6+5+2 = 15
```

```
 INPUT: Enter digits: 654321
        Enter sum: 20

OUTPUT: 6+5+3+2 = 20
        6+5+4+1 = 20
        6+4+3+2+1 = 20
```

(Note: The three lines may appear in a different order.)

3.4 Write a program to decompose a large positive integer into its prime factors. The input number will be less than 80 digits with all its prime factors less than 100. The prime factors must be displayed in ascending order, each followed by the power symbol (^) and the number of times the prime divides the number. If a prime divides the number only once, then do not display the power. Examples:

INPUT: Enter number: **1063382396627932698323045648224275608**

OUTPUT: **2^123**

INPUT: Enter number: **735154582240826037290332800000000000**

OUTPUT: **2^15 * 3^4 * 5^10 * 13^2 * 37 * 89^6 * 97^3**

3.5 Write a program to find words, associated with computers, within the following 12 X 11 array of letters:

```
                        1 1
      1 2 3 4 5 6 7 8 9 0 1

 1    D A T A A D F B A A M
 2    J A R B J C E D F O I
 3    R E A E E X E V D B C
 4    J E S U S D E E R N R
 5    F A B U U N M I E M O
 6    L L M N S O I P T K C
 7    P O Q R S I T R U O H
 8    A B U V K W S X P P I
 9    S O Y Z C P U L M L P
10    C C I S A B C D O A M
11    A E F G R H I J C R M
12    L K L E T T E K S I D
```

Words may appear vertically or horizontally, forward or backward. Display the coordinates of the first letter and the last letter of a word given as input. Examples:

INPUT: Enter word: **COMPUTER**
OUTPUT: **FIRST LETTER: (11, 9)**
LAST LETTER: (4, 9)

INPUT: Enter word: **CHIP**
OUTPUT: **FIRST LETTER: (6, 11)**
LAST LETTER: (9, 11)

INPUT: Enter word: **DISKETTE**
OUTPUT: **FIRST LETTER: (12, 11)**
LAST LETTER: (12, 4)

3.6 Write a program to display all integer values of X that satisfy two equations joined by either the logical operator **AND** or **OR**, where each equation is of the form: X>#, or X<# (where # is a single digit whole number).

- If no integers solve the equations, display: **NO SOLUTION**
- If all integers solve the equations display: **ALL INTEGERS**
- If there is a finite solution:
  - If there are six or less integers that solve the equations then list all the integers in ascending order separated by commas; otherwise list the first three integers separated by commas and then three periods and then the last three integers.
- If there are an infinite number of positive integers that solve the equations, then display the smallest three integers separated by commas, and then three periods.
- If there are infinite number of negative integers that solve the equations, then display three periods followed by the three largest numbers that satisfy the equations, separated by commas.
- If there are both an infinite number of positive solutions and negative solutions, but there is a gap between them, then display three spaces between these two infinite solutions as directed above.

Examples:

```
      INPUT: Enter equation 1: X>3
             Enter logical op: AND
             Enter equation 2: X<9
    OUTPUT: 4,5,6,7,8

     INPUT: Enter equation 1: X>3
            Enter logical op: OR
            Enter equation 2: X<0
    OUTPUT: ...-3,-2,-1   4,5,6...

      INPUT: Enter equation 1: X<1
             Enter logical op: OR
             Enter equation 2: X>0
    OUTPUT: ALL INTEGERS

     INPUT: Enter equation 1: X<2
            Enter logical op: AND
            Enter equation 2: X<5
    OUTPUT: ...-1,0,1

     INPUT: Enter equation 1: X>1
            Enter logical op: AND
            Enter equation 2: X<9
    OUTPUT: 2,3,4...6,7,8
```

3.7 Write a program to accept as input two 3 x 3 matrices (with each element no larger than 2 digits in base 16) and prints the SUM and PRODUCT (first times the second). Each element in the two results are to be displayed in base 16 and are to be right justified within 6 columns. All the elements of first matrix are input before entering the elements of the second matrix. Each element at (ROW, COL) in the PRODUCT is obtained by summing the three products of paired elements from row number ROW in matrix 1 with the elements from column number COL in matrix 2. Example:

```
INPUT: Enter Mat1 (1,1): 1       Enter Mat2 (1,1): 1
       Enter Mat1 (1,2): 2       Enter Mat2 (1,2): E
       Enter Mat1 (1,3): 3       Enter Mat2 (1,3): F2
       Enter Mat1 (2,1): A1      Enter Mat2 (2,1): 2
       Enter Mat1 (2,2): B2      Enter Mat2 (2,2): 99
       Enter Mat1 (2,3): C3      Enter Mat2 (2,3): 8D
       Enter Mat1 (3,1): D4      Enter Mat2 (3,1): 3
       Enter Mat1 (3,2): E5      Enter Mat2 (3,2): FF
       Enter Mat1 (3,3): F6      Enter Mat2 (3,3): 9E

OUTPUT: SUM =     2    10    F5
               A3   14B   150
               D7   1E4   194

        PRODUCT =     E   43D   3E6
                  44E 1356D 17296
                  580 1897F 1DE5D
```

Note: Element 17296, located at (2,3) in PRODUCT, is derived by summing the three products: A1 * F2, B2 * 8D, and C3 * 9E.

3.8 Write a program to find all sets of three 3-digit primes composed of the digits 1 through 9 such that their sum consists of four distinct digits in order of magnitude. Output must be of the following format:

```
### + ### + ### = ####
```

where ### represents the primes displayed in increasing order, and #### represents their sum. The seven sets of primes are to be displayed in order of magnitude by the first prime and then the second prime (if two sets have the same first prime). Two of the seven solutions are displayed below. Example:

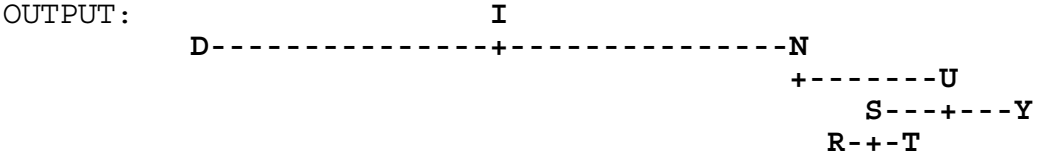
```
149 + 257 + 863 = 1269
### + ### + ### = ####
### + ### + ### = ####
241 + 367 + 859 = 1467
### + ### + ### = ####
### + ### + ### = ####
### + ### + ### = ####
```

3.9 A binary tree is a tree where each node has at most 2 children. A binary search tree has the value of each node greater than (or equal to) the value of its left child and less than the value of its right child.

Write a program to produce a binary search tree of letters from data input. Input will be one or more words separated by a space, but only the letters will be used in building the tree. No more than 9 row levels of letters will be produced from the data input. Examples:

INPUT: Enter word(s): **INDUSTRY**

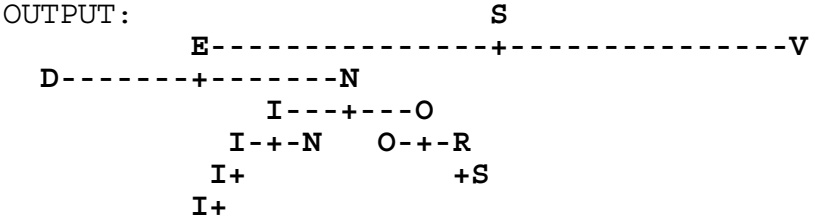
OUTPUT:



Note: In the above example, the first letter is centered. The second letter, N, is greater than I, so it is placed 16 positions to the right of I. The third letter, D, is less than I, so it is placed 16 positions to the left of I. The fourth letter, U, is greater than I and greater than N, so it is placed 8 positions to the right of N. The fifth letter, S, is greater than I and N, but less than U, so it is placed 4 positions to the left of U. The sixth letter, T, is greater than I and N, less than U, and greater than S, so it is placed 2 positions to the right of S. The seventh letter, R, is greater than I and N, but less than U and S, so it is placed 2 positions to the left of S. The last letter, Y, is greater than I, N, and U, so it is placed 4 positions to the right of U.

INPUT: Enter word(s): **SENIOR DIVISION**

OUTPUT:



Note: In the above example, the fourth occurrence of the letter I is placed 1 position to the left of the previous letter I. Every letter placed beyond the 5th level of nodes is placed either 1 position to the right or to the left of its parent.

3.10 Woolley's Paradox, published in the May 6, 1984 Miami Herald Neighbors section, "proves" that 2 = 4 by using parallel equations and the concept of infinity. A paradox is a self-contradictory statement that at first seems to be true. The fallacy of the statement presented in the newspaper can be seen in solving the following computer program.

Write a program to determine the range of values for which F(X) CONVERGES, as X approaches infinity for varying positive values of K. F(X) is defined below as a recursive function:

$$F(X) = \begin{cases} K & \text{if } X = 1 \\ K \wedge F(X-1) & \text{if } X > 1 \end{cases}$$

For K = 0.0,
as X approaches infinity, F(X) will equal 0.0 when X is odd and will equal 1.0 when X is even; therefore, F(X) does not converge for K = 0.0.

For K = 0.25,
as X approaches infinity, F(X) will converge on 0.5.

For K = 1.0,
as X approaches infinity, F(X) will equal 1.0.

For K = 2.0,
F(1) = 2.0,
F(2) = 2.0^F(1) = 4.0,
F(3) = 2.0^F(2) = 16.0,
F(4) = 2.0^F(3) = 65536.0, and
F(X) will diverge as X approaches infinity.

For practical purposes, the functional value at X = 5000 can be used for "the functional value as X approaches infinity," although it may not always be necessary to perform 5000 iterations to obtain the desired result.

Output must be of the form:

MINIMUM VALUE: F(X) = #.### **OCCURS WHEN K =** #.###
MAXIMUM VALUE: F(X) = #.# **OCCURS WHEN K =** #.#####

Note: #.### represents the actual minimum value F(X) takes on for some value of K. On the first line, both F(X) and K need to be rounded to the nearest 0.001.

Note: #.# represents the actual maximum value F(X) takes on for some value of K. F(X) must be rounded to the nearest 0.1. The value of K causing a maximum value for F(X) is represented by #.#####, and must be rounded to the nearest 0.00001.

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '94

1.1 Write a program to display the following lines, each beginning at the left most column of the screen:

FHSCC '94 IS SPONSORED BY:

GTEDS GTEDS GTEDS GTEDS GTEDS
GTEDS GTEDS GTEDS GTEDS GTEDS
GTEDS GTEDS GTEDS GTEDS GTEDS
GTEDS GTEDS GTEDS GTEDS GTEDS

USF CENTER FOR EXCELLENCE
USF CENTER FOR EXCELLENCE
USF CENTER FOR EXCELLENCE
USF CENTER FOR EXCELLENCE

FLORIDA DEPARTMENT OF EDUCATION
FLORIDA DEPARTMENT OF EDUCATION
FLORIDA DEPARTMENT OF EDUCATION
FLORIDA DEPARTMENT OF EDUCATION

1.2 Every couple of months, GTE Data Services (GTEDS) hires qualified individuals for their New Recruit Development program. This program features 11 and 13 week curriculums designed to train programmers in the standards and procedures of GTE Data Services. The course is "pass/fail" and successful students are placed in programmer positions in the company with a competitive starting salary. Entrance into the program is very competitive since GTEDS has available a pool of 300 potential candidates each period from which they select 15 individuals by reviewing resumes, conducting telephone interviews, administering aptitude tests, and holding panel interviews. If an applicant passes all the entrance requirements, he/she is offered a job with GTE Data Services.

Write a program to accept as input whether an applicant has PASSED or FAILED the entrance requirements and whether the applicant plans to ACCEPT or REJECT an offer if proposed, then display whether or not the applicant will be HIRED. Examples:

INPUT: Entrance requirement: **PASSED**
Plans to accept or reject offer: **ACCEPT**

OUTPUT: **APPLICANT WILL BE HIRED**

INPUT: Entrance requirement: **FAILED**
Plans to accept or reject offer: **REJECT**

OUTPUT: **APPLICANT WILL NOT BE HIRED**

1.3 GTE is the largest U.S.-based local-telephone company with domestic and international operations providing services in 40 states and 70 countries. GTE is the second largest U.S. cellular provider, the third largest U.S. data processing company, and the fourth largest publicly-owned telecommunications company in the world.

Write a program to display the total number of employees who will be working at GTE after accepting as input the current number of employees, the number of employees that are about to be hired, and the number of employees who are about to retire or leave. The total number of employees displayed will be six digits long. Example:

```
 INPUT: Enter current number: 131000
        Enter number hiring: 943
        Enter number leaving: 431

OUTPUT: 131512 EMPLOYEES
```

1.4 The Customer Billing Services System (CBSS) is an advanced and highly flexible customer billing system that produces easy-to-read and understandable-- yet detailed-- bills. GTE is in the process of converting from the age-old Customer Record Billing (CRB) system to CBSS. Although the Conversion group under Bonnie's supervision and many others have worked hard to convert several regions to CBSS, the six 1993 conversions of approximately 7.3 million customer accounts were seen as "non-events" in the eyes of others since they went so smoothly.

Write a program to accept as input several quantities of customer accounts for a set of regions that were converted, and then display the total number of customer accounts converted to CBSS. Input/Output will be of the following formats, depending on whether the quantity is integral or contains a decimal:

```
# MILLION    or    #.# MILLION
```

Input will be terminated by entering -999. If the output is integral, then the number must be displayed without the decimal. Example:

```
 INPUT: Enter number of accounts: 1 MILLION
        Enter number of accounts: 1.8 MILLION
        Enter number of accounts: 1.2 MILLION
        Enter number of accounts: 1.3 MILLION
        Enter number of accounts: 0.5 MILLION
        Enter number of accounts: 1.5 MILLION
        Enter number of accounts: -999

OUTPUT: 7.3 MILLION ACCOUNTS CONVERTED TO CBSS
```

1.5 A student worked during the summer at a company that paid time and a half for each hour worked over 40 hours per week. Write a program to compute the gross wages the student earns given the hours worked and the hourly rate as input. The gross wages computed will not exceed $999.99 and will not be less than $100.00. Examples:

INPUT: Enter hours, rate: **33, 5.25**

OUTPUT: **GROSS WAGES ARE $173.25**

INPUT: Enter hours, rate: **45, 6**

OUTPUT: **GROSS WAGES ARE $285.00**

1.6 GTE has announced its strategic effort to trade or sell a small percentage of local-exchange properties in markets that may be of greater long-term strategic value to other telephone-service providers. GTE may sell its customer access lines based on the area code of service. When this happens, all the data is copied from GTE's databases for selected accounts and sent to another company on tape. All data on the databases associated with the accounts in the designated area codes may be deleted after this extraction. Ralph is a project leader in the Conversion/Repositioning group at GTEDS and would like to know the number of accounts that will be repositioned in several areas.

Write a program to use the following set of area codes with its corresponding fictitious number of accounts, and accept as input several of these area codes that will be sold to another company, and then display the total number of accounts being sold:

Area Code	Accounts
706	95000
208	54321
912	99825
605	88776
404	90175

Example:

INPUT: Enter number of area codes: **3**
Enter area code: **912**
Enter area code: **706**
Enter area code: **404**

OUTPUT: **TOTAL NUMBER OF ACCOUNTS BEING SOLD = 285000**

1.7 Software quality can be improved by structured testing, and development costs can be greatly reduced by catching errors or defects early. Software errors cost more to fix as the development cycle advances. Flaws not identified until after installation are corrected in the maintenance cycle. According to the National Bureau of Standards, the cost to fix a line of code in the maintenance cycle is 80 times the cost it would have been if the error was found in the new development cycle.

Write a program to display the cost involved in fixing a problem in a given phase, where the cost to fix the problem in the REQUIREMENTS phase is given as input along with the phase in which the problem was found. All costs will be integers less than 32,000. The following list shows the multiplication factor of the cost of fixing a problem in a phase in comparison to the requirements phase:

Phase	Factor
Requirements	1x
Design	5x
Coding	10x
System Test	20x
Acceptance Test	50x
Maintenance	100x

Examples:

INPUT: Enter cost $: **66**
Enter phase: **MAINTENANCE**

OUTPUT: **COST IS $6600 TO FIX PROBLEM IN MAINTENANCE PHASE**

INPUT: Enter cost $: **65**
Enter phase: **ACCEPTANCE TEST**

OUTPUT: **COST IS $3250 TO FIX PROBLEM IN ACCEPTANCE TEST PHASE**

INPUT: Enter cost $: **99**
Enter phase: **REQUIREMENTS**

OUTPUT: **COST IS $99 TO FIX PROBLEM IN REQUIREMENTS PHASE**

1.8 GTEDS has a project in Commercial Services that processes many Medicare Part-B claims nationwide. The Medicare project holds a major part in the national health care market holding contracts with 13 states, including Florida. One month the number of claims processed in Florida exceeded 3.5 million, which took an incredible amount of DASD (direct access storage device), also known as disk space. Maintaining this voluminous processing takes maximum efficiency efforts on the part of programming personnel. Maximum utilization of blocksizes are needed for hundreds of different files. In JCL (job control language) the blocksize is computed as the largest multiple of the logical record length that is less than half of a track length on a 3380 IBM disk pack, which amounts to 23,476 bytes.

Write a program to calculate the maximum blocksize on a 3380 IBM disk pack given the logical record length as input. Examples:

INPUT: Enter logical record length: **80**

OUTPUT: **BLOCKSIZE = 23440 BYTES**

INPUT: Enter logical record length: **100**

OUTPUT: **BLOCKSIZE = 23400 BYTES**

1.9 The electric company charges $5.65 per kilowatt hour. Customers who use less than 10 kilowatt hours per month receive a discount rate of $4.95. Each customer must also pay a 3% utility tax and a 6% state tax. Each tax is computed on the product of the charge and the number of hours. Customers who use more than 30 kilowatt hours per month pay a non-taxable $25 service fee.

Write a program to compute a customer's bill rounded to the nearest penny, given as input the number of kilowatt hours the customer used (as a real number less than 100). Examples:

INPUT: Enter kilowatt hours: **11.5**

OUTPUT: **THE CUSTOMER'S ELECTRIC BILL IS $70.82**

INPUT: Enter kilowatt hours: **9**

OUTPUT: **THE CUSTOMER'S ELECTRIC BILL IS $48.56**

INPUT: Enter kilowatt hours: **35**

OUTPUT: **THE CUSTOMER'S ELECTRIC BILL IS $240.55**

1.10 A matrix is symmetric if when you fold it along the major diagonal the entries above this diagonal exactly match the entries below this diagonal. The major diagonal extends from the top left to the bottom right of the matrix.

Write a program to determine if a given 5 by 5 matrix is symmetric. Each element will be a single digit. Examples:

```
 INPUT: Enter row: 1, 5, 6, 7, 8
        Enter row: 5, 2, 7, 7, 9
        Enter row: 6, 7, 3, 4, 0
        Enter row: 7, 7, 4, 4, 1
        Enter row: 8, 9, 0, 1, 5

OUTPUT: MATRIX IS SYMMETRIC


 INPUT: Enter row: 1, 2, 3, 4, 5
        Enter row: 5, 4, 3, 2, 1
        Enter row: 1, 2, 3, 4, 5
        Enter row: 6, 7, 8, 9, 0
        Enter row: 0, 9, 8, 7, 6

OUTPUT: MATRIX IS NOT SYMMETRIC
```

2.1 The National Test Facility at GTEDS uses the utility ESP to submit jobs in a set processing order to test the functionality of the Customer Billing Services System (CBSS). Greg, a supervisor within NTF, insists that his test technicians thoroughly examine the jobs run between check points before proceeding to run the next set of jobs. If a problem is encountered within a job, one or more of the jobs may need to be re-run.

Write a program to simulate the ESP environment. The program is to accept as input a string of two character jobs and check points, each separated by a space. The jobs are to be run in the order given. A check point, indicated by a CK, may succeed a set of jobs. The program then displays one job per line and prompts the technician at the check point EVERYTHING OK? If 'N' is the response, then re-display all the jobs just displayed since the previous check point, or the start of the sequence--if this is the first check point. If 'Y' is the response, then display the next set of jobs to run until a check point is encountered and there are no other jobs to run. Example:

```
 INPUT: Enter jobs/CK: UH UN UB CK UD UI CK UR CK

OUTPUT: UH
        UN
        UB
        EVERYTHING OK?
 INPUT: N
OUTPUT: UH
        UN
        UB
        EVERYTHING OK?
 INPUT: Y
OUTPUT: UD
        UI
        EVERYTHING OK?
 INPUT: Y
OUTPUT: UR
        EVERYTHING OK?
 INPUT: N
OUTPUT: UR
         EVERYTHING OK?
 INPUT: Y
OUTPUT: (program ends)
```

2.2 Write a program to clear the screen, randomly choose a letter, and display this letter in random locations until a key is pressed. If a letter is pressed, the program clears the screen and displays the selected letter in random locations until a key is pressed; if the space bar is pressed, the screen is cleared and a random letter is chosen and displayed in random locations until a key is pressed; if any other key is pressed, the program terminates. Have the program display the letters slowly (about 10 per second). Example:

RUN PROGRAM:

OUTPUT: (The screen is cleared and a randomly chosen letter is displayed in random locations until a key is pressed, displaying the letters slowly (about 10 per second))
INPUT: Enter letter: **R**
OUTPUT: (The program clears the screen and continuously displays the letter **R** in random locations until a key is pressed)
INPUT: Enter letter: **C**
OUTPUT: (The program clears the screen and continuously displays the letter **C** in random locations until a key is pressed)
INPUT: Enter letter: (press space bar)
OUTPUT: (The program clears the screen and displays a randomly chosen letter in random locations until a key is pressed)
INPUT: Enter letter: **Y**
OUTPUT: (The program clears the screen and continuously displays the letter **Y** in random locations until a key is pressed)
INPUT: Enter letter: (press space bar)
OUTPUT: (The program clears the screen and displays a randomly chosen letter in random locations until a key is pressed)
INPUT: Enter letter: (press space bar)
OUTPUT: (The program clears the screen and displays a randomly chosen letter in random locations until a key is pressed)
INPUT: Enter letter: **3**
OUTPUT: (program terminates)

2.3 The Hebrew alphabet consists of twenty-two consonants. The name of each letter is displayed below with its English transliteration.

Name	Transliteration
ALEPH	)
BETH	B
GIMEL	G
DALETH	D
HE	H
WAW	W
ZAYIN	Z
HETH	CH
TETH	T
YOD	Y
KAPH	K
LAMED	L
MEM	M
NUN	N
SAMEKH	S
AYIN	(
PE	P
TSADHE	TS
QOPH	Q
RESH	R
SIN	S
TAW	T

Write a program to convert a set of Hebrew letters into its English transliteration. Hebrew is read from right to left and transliterated from left to right. No more than 6 letter names will be entered, each separated by one space.

Examples:

INPUT: Enter letters: **LAMED LAMED HETH MEM**

OUTPUT: **MCHLL**

INPUT: Enter letters: **AYIN NUN TSADHE HE WAW**

OUTPUT: **WHTSN(**

INPUT: Enter letters: **HE WAW HE YOD**

OUTPUT: **YHWH**

2.4 Megabyte Bank & Trust is issuing both visa and mastercards. For security reasons, the membership department needs a program that will compute the sum of the individual digits in the account number to see if the total is even or odd. To ensure that the sum of all numbers is odd, a 1 will be appended to the end of the even totals, and a 0 to the end of the odd totals. Before the "security digit" is added, a valid account number contains 7 or 9 digits.

Write a program to accept as input an account number, and then append a "security digit" if the account number is valid.
If the account number is not 7 or 9 characters long, then display: ERROR - INCORRECT LENGTH.
If the account number has a non-numeric character, then display: ERROR - NON-NUMERIC.

Examples:

INPUT: Enter account number: **23098491**

OUTPUT: **ERROR - INCORRECT LENGTH**

INPUT: Enter account number: **2098123**

OUTPUT: **20981230**

INPUT: Enter account number: **309812300**

OUTPUT: **3098123001**

INPUT: Enter account number: **234498A**

OUTPUT: **ERROR - NON-NUMERIC**

INPUT: Enter account number: **ABC**

OUTPUT: **ERROR - INCORRECT LENGTH**
ERROR - NON-NUMERIC

2.5 Write a program to display the number of times each of the digits 0 through 9 are used in the page numbers of a book given the last page number as input. Each page is numbered except for the first page of every chapter. Unnumbered pages are page 1 and every page divisible by M, where M is given as input. The program must also display those digits appearing the most and the least, with each set of number(s) in ascending order and separated by a space (if more than one). Example:

```
 INPUT: Enter last page number: 2345
        Enter M: 23

OUTPUT: 0 APPEARS 635 TIMES
        1 APPEARS 1698 TIMES
        2 APPEARS 1072 TIMES
        3 APPEARS 690 TIMES
        4 APPEARS 642 TIMES
        5 APPEARS 636 TIMES
        6 APPEARS 636 TIMES
        7 APPEARS 635 TIMES
        8 APPEARS 635 TIMES
        9 APPEARS 636 TIMES

        DIGIT(S) APPEARING THE MOST: 1
        DIGIT(S) APPEARING THE LEAST: 0 7 8
```

2.6 For extra credit in an algebra class, a student could write a program to list the nature of the roots and compute the roots for any quadratic equation in standard form (AX^2 + BX + C = 0) when the coefficients A, B, and C are entered into the program.

Write a program to accept the coefficients of a quadratic equation and display if the nature of the roots is REAL or COMPLEX and then display the root(s) of the quadratic equation in descending order. Numbers input and output will be integers, and 'I' equals the square root of -1. Examples:

```
 INPUT: Enter coefficients A, B, C: 1, -5, 6

OUTPUT: THE ROOTS ARE REAL
        THE ROOTS ARE 3 AND 2


 INPUT: Enter coefficients A, B, C: 1, 4, 4

OUTPUT: THE ROOTS ARE REAL
        THE ONLY ROOT IS -2


 INPUT: Enter coefficients A, B, C: 1, 4, 8

OUTPUT: THE ROOTS ARE COMPLEX
        THE ROOTS ARE -2 + 2I AND -2 - 2I
```

2.7 A customer account number is a system-generated 10-digit number assigned to a customer who may have one or more services. Each customer account number is generated from the last 9-digit seed number used.

Write a program to generate the next 15 customer account numbers given a seed number, less than 10 digits, as input. Use the following algorithm for generating each customer account number:

```
Retrieve seed number used last:                          10001235
Add 1 to seed number to generate new seed number:        10001236
Pad front with zeroes to make 9 digits (if needed):    010001236
Reverse the last two digits:                           010001263
Shift digits 3-9 two positions to the right:           01  0001263
Move last 2 digits to positions 3-4:                   016300012
Calculate and add check digit                          0163000123
```

by summing the following products:

- multiply the first digit by 10
- multiply the second digit by 9
- multiply the third digit by 8
- multiply the fourth digit by 7
- multiply the fifth digit by 6
- multiply the sixth digit by 5
- multiply the seventh digit by 4
- multiply the eighth digit by 3
- multiply the ninth digit by 2;

then divide the sum by 11 and subtract the resulting remainder from 11 for the check digit.
In this case, the sum is 85 = (1*9 + 6*8 + 3*7 + 1*3 + 2*2) and 85 / 11 = 7 remainder 8. The check digit is 11 - 8 = 3.
If the check digit is 11, then it becomes 0. A check digit of 10 is invalid, and a new customer number is generated using the next seed number.

Example:

```
INPUT: Enter seed used last: 1133126

OUTPUT: 0072113316
        0082113319
        0092113311
        0003113310
        0013113313
        0023113316
        0033113319
        0043113311
        0053113314
        0063113317
        0083113312
        0093113315
        0004113314
        0014113317
        0034113312
```

2.8 One of the basic fundamentals of navigation is the art of dead-reckoning (DR). A DR position is calculated using two of three basic pieces of information: distance, time, speed. Thus, the progress of a trip (by boat, car, or plane) can be kept by plotting each position on a map. For example, if you are traveling from Tampa to New York City by plane, given the speed of the airplane and tracking the time since take-off, you can estimate the progress of the plane on a map as you fly along.

Write a program to accept as input three values: speed, distance, and time. The program will then calculate the unknown value (entered as 0) using the other two values. Speed will be entered as a real number in mph, and distance will be entered as a real number in miles. Time can be entered in one of three formats: minutes (i.e. 45M), decimal hours (i.e. 1.34H), or clock hours/minutes (i.e. 02:23C). Display any speed and distance rounded to one decimal place, and display time rounded to the second decimal point according to the following formats:

```
SPEED = nnn.n MPH
DISTANCE = nnnn.n MILES
TIME = n.nn HOURS
```

Examples:

```
 INPUT: Enter speed, distance: 60.1, 0
        Enter time: 01:15C

OUTPUT: DISTANCE =   75.1 MILES


 INPUT: Enter speed, distance: 75.0, 93.1
        Enter time: 0

OUTPUT: TIME = 1.24 HOURS


 INPUT: Enter speed, distance: 0, 25.0
        Enter time: 150M

OUTPUT: SPEED =  10.0 MPH


 INPUT: Enter speed, distance: 0, 70.0
        Enter time: 3.5H

OUTPUT: SPEED =  20.0 MPH
```

2.9 Since customer satisfaction is a goal at GTEDS, response times to the customer are calculated and monitored. The response time begins when the customer "reports" an incident and ends when the incident has been "cleared."

Write a program to accept as input a "reported" date and time and a "cleared" date and time, and then display the response time in minutes. Both dates will be given in the form mm/dd/yy with the same month and year, and the time will be given in the form hh:mm. Only time between working hours (8 a.m. and 5 p.m.) are to be computed in the response time. Examples:

```
 INPUT: Enter reported date: 01/17/94
        Enter reported time: 08:05
        Enter cleared date: 01/19/94
        Enter cleared time: 18:15

OUTPUT: RESPONSE TIME WAS 1615 MINUTES


 INPUT: Enter reported date: 02/03/94
        Enter reported time: 05:35
        Enter cleared date: 02/03/94
        Enter cleared time: 14:25

OUTPUT: RESPONSE TIME WAS 385 MINUTES


 INPUT: Enter reported date: 12/05/94
        Enter reported time: 22:44
        Enter cleared date: 12/16/94
        Enter cleared time: 01:23

OUTPUT: RESPONSE TIME WAS 5400 MINUTES
```

2.10 The GTEDS Customer Billing Services System (CBSS) has a group responsible for re-rating long distance calls based on pricing plans such as AT&T's Reachout America plan. Given the criteria shown below, write a program to accept five inputs (shown in the examples) and determine for which pricing plan(s) the person qualifies and what the new cost would be for the call (less than $100). A person will receive only one discount. The plan received (and displayed) will be the greatest discount applicable. Partial cents are rounded for the final charges.

```
          Plan A Criteria:
               Call lasts for at least 5 minutes.
               Call is to another area code.
               15% discount is applied.
          Plan B Criteria:
               Person has a physical handicap.
               10% discount is applied.
          Plan C Criteria:
               Call is to area code 407.
               Call is to another area code.
               Call lasts for at least 3.5 minutes.
               12.25% discount is applied.
```

Examples:

```
     INPUT: Enter originating number: 8135558530
            Enter number called: 3055551212
            Handicapped person?: YES
            Enter length of call: 8
            Enter cost of call $: 5.42

    OUTPUT: THE PLAN A CHARGE WOULD BE $4.61
            THE PLAN B CHARGE WOULD BE $4.88
            THIS PERSON WOULD RECEIVE PLAN A


     INPUT: Enter originating number: 8135558530
            Enter number called: 4075551212
            Handicapped person?: NO
            Enter length of call: 3
            Enter cost of call $: 5.42

    OUTPUT: THIS PERSON DOES NOT QUALIFY FOR ANY PLANS


     INPUT: Enter originating number: 8135558530
            Enter number called: 4075551212
            Handicapped person?: YES
            Enter length of call: 15
            Enter cost of call $: 11.44

    OUTPUT: THE PLAN A CHARGE WOULD BE $9.72
            THE PLAN B CHARGE WOULD BE $10.30
            THE PLAN C CHARGE WOULD BE $10.04
            THIS PERSON WOULD RECEIVE PLAN A
```

3.1 The Greek alphabet consists of twenty-four letters. Every dormitory on the University of South Florida campus is named after a Greek letter. Many honor societies are named after Greek letters (Mu Alpha Theta). The name of each letter is displayed below with its English transliteration and its numerical value.

Name	Transliteration	Numerical Value
ALPHA	A	1
BETA	B	2
GAMMA	G	3
DELTA	D	4
EPSILON	E	5
ZETA	Z	7
ETA	E	8
THETA	TH	9
IOTA	I	10
KAPPA	K	20
LAMBDA	L	30
MU	M	40
NU	N	50
XI	X	60
OMICRON	O	70
PI	P	80
RHO	R	100
SIGMA	S	200
TAU	T	300
UPSILON	U	400
PHI	PH	500
CHI	CH	600
PSI	PS	700
OMEGA	O	800

Write a program to convert an English transliteration of a Greek word to the names of the Greek letters composing the word, and display the sum of the numerical values for each of the letters. For the purpose of this program:

1) The letter combinations TH, PH, and PS; will take precedence over the individual letters T, P, and S, when converting;
2) E will become EPSILON (not ETA), and O will become OMEGA (not OMICRON);
3) The transliteration will have at most 13 letters.

Examples:

INPUT: Enter transliteration: **PHILANTHROPIA**
OUTPUT: **PHI IOTA LAMBDA ALPHA NU THETA RHO OMEGA PI IOTA ALPHA**
NUMERICAL SUM = 1591

INPUT: Enter transliteration: **CHARISMATA**
OUTPUT: **CHI ALPHA RHO IOTA SIGMA MU ALPHA TAU ALPHA**
NUMERICAL SUM = 1253

3.2 A taxi driver is in a city whose north/south streets are numbered from 1 (northernmost) through 8 (southernmost), and the east/west streets are represented by the letters of the alphabet A (farthest west) through Z (farthest east). The taxi is given a starting point at a north/south, east/west intersection (such as 5,T). The taxi may never be more than two city blocks from the starting point and may not exit the city limits. The boundary for a taxi starting at position 4,D is shown by the + below:

```
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
1
2   + + + + +
3   +       +
4   +   *   +
5   +       +
6   + + + + +
7
8
```

Write a program that accepts as input a starting point, and then will accept a series of directions (N, S, E, W) or Q to quit, and display the new position of a taxi if a valid move is made. If the attempted move would cause the taxi to exit the city limits or move outside the two block limit, the taxi is left where it is and an appropriate warning is issued among the following:

```
LOCATION IS OUTSIDE CITY LIMITS
LOCATION IS TOO FAR NORTH
LOCATION IS TOO FAR SOUTH
LOCATION IS TOO FAR EAST
LOCATION IS TOO FAR WEST
```

Example:

```
 INPUT: Enter starting position: E,2
        Enter direction: N
OUTPUT: TAXI LOCATION IS E,1
 INPUT: Enter direction: N
OUTPUT: LOCATION IS OUTSIDE CITY LIMITS
 INPUT: Enter direction: E
OUTPUT: TAXI LOCATION IS F,1
 INPUT: Enter direction: E
OUTPUT: TAXI LOCATION IS G,1
 INPUT: Enter direction: E
OUTPUT: LOCATION IS TOO FAR EAST
 INPUT: Enter direction: S
OUTPUT: TAXI LOCATION IS G,2
 INPUT: Enter direction: S
OUTPUT: TAXI LOCATION IS G,3
 INPUT: Enter direction: S
OUTPUT: TAXI LOCATION IS G,4
 INPUT: Enter direction: S
OUTPUT: LOCATION IS TOO FAR SOUTH
 INPUT: Enter direction: Q
OUTPUT: (program terminates)
```

3.3 An anagram is a word that contains the same letters as another word, but in a different order. For example, DARE and READ are anagrams of each other.

Write a program to accept as input a list of words and then display all pairs of anagrams in alphabetical order within the pair and among the pairs (based on the first word of the pair). If no anagrams are found then display: NO ANAGRAMS IN LIST. Less than 10 words will be entered, each word having less than 8 letters. Examples:

```
 INPUT: Enter number of words: 4
        Enter word: WORD
        Enter word: DRAW
        Enter word: WARD
        Enter word: WIND

OUTPUT: ANAGRAMS: DRAW, WARD
```

```
 INPUT: Enter number of words: 7
        Enter word: READ
        Enter word: WARD
        Enter word: DARE
        Enter word: EAR
        Enter word: TAP
        Enter word: PAT
        Enter word: DRAW

OUTPUT: ANAGRAMS: DARE, READ
                  DRAW, WARD
                  PAT, TAP
```

```
 INPUT: Enter number of words: 3
        Enter word: LEAF
        Enter word: FEAR
        Enter word: EAR

OUTPUT: NO ANAGRAMS IN LIST
```

3.4 Mike posed an interesting problem to Doug during some lax time within their busy work schedules. Suppose you have $30 in 4 envelopes, and you take some money from the envelope with the most money and disperse it in the other envelopes so that the 4 envelopes have the same amount as before, but in a different order. Mike asked "how much is in each envelope?" After thinking, Doug responded with 6, 7, 8, and 9 since 3 dollars could be taken from 9 making 7, 8, 9, and 6. Mike then realized that there is more than one solution to this problem since he was thinking of the solution 2, 4, 8, and 16 since 14 dollars could be taken from 16 making 4, 8, 16, and 2.

Write a program to find all solutions to this envelope problem for an amount of money given as input (between 10 and 30 dollars inclusive). Display each solution in ascending order with respect to the initial amounts in each envelope. Display each solution in ascending order among the other solutions. Examples:

INPUT: Enter amount of money: **14**

OUTPUT:
```
TAKE 1 2 3 8 AND DISPERSE 7 DOLLARS TO MAKE 2 3 8 1
TAKE 1 2 4 7 AND DISPERSE 6 DOLLARS TO MAKE 2 4 7 1
TAKE 1 2 5 6 AND DISPERSE 5 DOLLARS TO MAKE 2 5 6 1
TAKE 1 3 4 6 AND DISPERSE 5 DOLLARS TO MAKE 3 4 6 1
TAKE 2 3 4 5 AND DISPERSE 3 DOLLARS TO MAKE 3 4 5 2
TOTAL NUMBER OF SOLUTIONS = 5
```

INPUT: Enter amount of money: **20**

OUTPUT:
```
TAKE 1 2 3 14 AND DISPERSE 13 DOLLARS TO MAKE 2 3 14 1
TAKE 1 2 4 13 AND DISPERSE 12 DOLLARS TO MAKE 2 4 13 1
TAKE 1 2 5 12 AND DISPERSE 11 DOLLARS TO MAKE 2 5 12 1
TAKE 1 2 6 11 AND DISPERSE 10 DOLLARS TO MAKE 2 6 11 1
TAKE 1 2 7 10 AND DISPERSE 9 DOLLARS TO MAKE 2 7 10 1
TAKE 1 2 8 9 AND DISPERSE 8 DOLLARS TO MAKE 2 8 9 1
TAKE 1 3 4 12 AND DISPERSE 11 DOLLARS TO MAKE 3 4 12 1
TAKE 1 3 5 11 AND DISPERSE 10 DOLLARS TO MAKE 3 5 11 1
TAKE 1 3 6 10 AND DISPERSE 9 DOLLARS TO MAKE 3 6 10 1
TAKE 1 3 7 9 AND DISPERSE 8 DOLLARS TO MAKE 3 7 9 1
TAKE 1 4 5 10 AND DISPERSE 9 DOLLARS TO MAKE 4 5 10 1
TAKE 1 4 6 9 AND DISPERSE 8 DOLLARS TO MAKE 4 6 9 1
TAKE 1 4 7 8 AND DISPERSE 7 DOLLARS TO MAKE 4 7 8 1
TAKE 1 5 6 8 AND DISPERSE 7 DOLLARS TO MAKE 5 6 8 1
TAKE 2 3 4 11 AND DISPERSE 9 DOLLARS TO MAKE 3 4 11 2
TAKE 2 3 5 10 AND DISPERSE 8 DOLLARS TO MAKE 3 5 10 2
TAKE 2 3 6 9 AND DISPERSE 7 DOLLARS TO MAKE 3 6 9 2
TAKE 2 3 7 8 AND DISPERSE 6 DOLLARS TO MAKE 3 7 8 2
TAKE 2 4 5 9 AND DISPERSE 7 DOLLARS TO MAKE 4 5 9 2
TAKE 2 4 6 8 AND DISPERSE 6 DOLLARS TO MAKE 4 6 8 2
TAKE 2 5 6 7 AND DISPERSE 5 DOLLARS TO MAKE 5 6 7 2
TAKE 3 4 5 8 AND DISPERSE 5 DOLLARS TO MAKE 4 5 8 3
TAKE 3 4 6 7 AND DISPERSE 4 DOLLARS TO MAKE 4 6 7 3
TOTAL NUMBER OF SOLUTIONS = 23
```

3.5 Often there is a need to save space by storing information in as little space as possible at GTEDS. One small way of accomplishing this is by using a shorter form of the regular Gregorian date (ex. mm/dd/yy, 02/03/94) in a date form called Julian (ex. yyddd, 94034 = 02/03/94). Functions are called in many programs at GTEDS to convert this Julian date to a Gregorian date and vise versa.

Write a program to convert a Gregorian date to Julian or a Julian date to a Gregorian date. Examples:

```
 INPUT: Enter Gregorian or Julian: GREGORIAN
        Enter date: 02/29/92
OUTPUT: JULIAN DATE = 92060

 INPUT: Enter Gregorian or Julian: JULIAN
        Enter date: 96366
OUTPUT: GREGORIAN DATE = 12/31/96
```

3.6 The GTEDS CBSS system has a Windows based application which allows their customer to paint a phone bill on the screen and attach logic to it. This information is saved as a 4GL and is then converted to COBOL. There is a program which does error checking on the 4GL and it is very important that the syntax is correct for this 4GL file. GTEDS wants to ensure that the Windows application was used to create or modify the 4GL file. They have developed a program which generates a key for this purpose and writes it to that file. The program which converts the 4GL to COBOL then reads this key to ensure that the file was created or modified by the Windows application. This key is derived by taking the number of bytes in a program and converting that number from one base to another and then reversing the resulting number.

Write a program that accepts as input a number of bytes and will convert that number from one base to another and then reverse the resulting number, given the two bases as input. The number input will be less than 10 million (equivalent in base 10), and each base will be between 2 and 16 inclusive. Examples:

```
 INPUT: Enter base of first number: 10
        Enter number: 65432
        Enter base of output: 16

OUTPUT: 89FF


 INPUT: Enter base of first number: 16
        Enter number: 98FF
        Enter base of output: 8

OUTPUT: 773411
```

3.7 Byron is a project leader for CBSS Conversion, and he would like to sort a stack of resolved IR (Incident Record) documents in ascending order according to the IR number.

Write an efficient program to sort 8,000 IR numbers generated by the formula shown below using a seed number given as input, and then display every 1000th number in ascending order within the sorted list. The following congruential "random" number generator is to be used to generate the 8,000 numbers to be sorted:

X(n) = 69069 * X(n - 1) mod 2^**20**

The first number in the list, X(1), is generated using the seed number input, X(0). The second number in the list, X(2), is generated by the previous number, X(1). The symbol, ^, means "raised to the power of". In order to solve this program efficiently, a list of relative running times are shown below for several sorting algorithms that sorted 8,000 integer elements on an IBM PC/AT-class machine:

Algorithm	Time
Quicksort	2.6 sec
Shellsort	4.5 sec
Mergesort	5.1 sec
Treesort	6.0 sec
Heapsort	6.6 sec
Insertion Sort	4.8 min
Selection Sort	10.0 min
Bubble Sort	15.0 min

The above times will be about 3 times faster on an 80386-class or 80486-class PC. However, the corresponding algorithms will take about 5 times longer to sort the REAL whole numbers generated by the congruential formula stated above. The **quicksort** is the fastest of all known comparison sorts for average data. The **bubblesort** is the slowest of any sorting algorithms, but is very popular since its logic is easy to remember. The **shellsort** is very quick and is one of best algorithms because it is iterative rather than recursive and therefore can be easily encoded in any language. The quicksort, bubblesort, and shellsort can sort 8,000 real numbers on an 80386-class PC in 5 seconds, 18 minutes, and 9 seconds respectively. Only one run (Input/Output) will be given for the judging criteria. Example:

INPUT: Enter seed X(0): **2345**

OUTPUT: **1000TH NUMBER = 130169**
2000TH NUMBER = 261293
3000TH NUMBER = 388629
4000TH NUMBER = 522973
5000TH NUMBER = 651533
6000TH NUMBER = 782081
7000TH NUMBER = 917085
8000TH NUMBER = 1048553

3.8 The ratio of the circumference of a circle to its diameter, represented by the Greek letter PI, has intrigued mathematicians and scientists. For thousands of years people have been trying to approximate as accurately as possible the value of this irrational number. PI was approximated as 3 in the Bible (1 Kings 7:23). Archimedes (287-212 B.C.) approximated PI to lie between 3 1/7 and 3 10/71. Ptolemy, in 150 A.D., estimated PI at 3.1416. Today, we are able to compute PI to hundreds of thousands of places. The value of PI correct to 110 decimal places is as follows:

```
PI = 3.14159265358979323846264338327950288419716939937510582097
       494459230781640628620899862803482534211706798214808651
```

Write a program to compute the volume of a sphere, accurate to N decimal digits, given the radius of the sphere as input. Both N and the radius will be input as whole numbers less than 101. The formula for computing the volume of a sphere is:

```
4/3 * PI * radius * radius * radius
```

Note: Be sure to handle accurately the quotient 4/3 in the above formula. The last decimal digit is truncated, not rounded. The output may wrap around the end of the screen.

Examples:

```
 INPUT: Enter N: 100
        Enter radius: 10

OUTPUT: 4188.79020478639098461685784437267051226289253250014109
        46332594564104218750482786648373797671228227573095


 INPUT: Enter N: 90
        Enter radius: 100

OUTPUT: 4188790.20478639098461685784437267051226289253250014109
        4633259456410421875048278664837379767122822


 INPUT: Enter N: 85
        Enter radius: 55

OUTPUT: 696909.97032133580006562972385750305647773874509471097
        46196085420602839394611573628623190587
```

3.9 The U.S. Postal Service requires large organizations and businesses sending a large volume of mail to use the new 11-digit bar codes if they want to take advantage of reduced rates for ZIP+4 mail and bar-coded mail. The new 11-digit bar codes are called Delivery Point Bar Codes (DPBC), and they allow machines to sort letter mail into the order a carrier would deliver it. Bar codes are the strings of vertical lines that appear near the address (usually below it) on an envelope. The following shows the bar code representation for each digit used in a DPBC:

```
   !!   ! !   !!   !  !  ! !   !!   !   ! !  !  ! !   !!
 !!!!! !!!!! !!!!! !!!!! !!!!! !!!!! !!!!! !!!!! !!!!! !!!!!

   1     2     3     4     5     6     7     8     9     0
```

Some bars are full size and others are smaller (half bars), designating "on" or "off". From left to right, the five bars for any given digit always are used to represent the values 7, 4, 2, 1, and 0. Only the full bars, the ones that are "on," count when read by a bar code reader, or by the eye. Each group of five bars representing a digit always uses two and only two full bars, and the values of those two bars are added together to give any digit from 1 to 9: 1=1+0, 2=2+0, 3=2+1, 4=4+0, 5=4+1, 6=4+2, 7=7+0, 8=7+1, 9=7+2. The digit 0 is a special case and is represented by a group of five bars with the two on the immediate left being full bars. An 11-digit DPBC consists of the ZIP+4 followed by the delivery point digits, taken from the last two digits of the street address or the P.O. BOX. For a DPBC, a 12th digit (called a check digit) is added to make the total of itself and the previous digits add to a number divisible by 10. Bar codes begin and end with single tall bars that are called framing bars, thus making a DPBC contain (1 + 11*5 + 1*5 + 1) = 62 bars.

Write a program to produce a Delivery Point Bar Code (DPBC), given two lines of input for the address. If the last 4 digits of a ZIP+4 are missing, then use 0000 when a street address is given, or use the last 4 digits of a P.O. BOX. Examples:

```
     INPUT: Enter address 1: 201 GTE ST
            Enter address 2: THISTOWN FL 12345-6789

    OUTPUT: DELIVERY POINT BAR CODE = 123456789014

!   !!  ! !  !!  !  ! ! !  !!  !   !!  ! ! !  !!      !! !  !!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


     INPUT: Enter address 1: P.O. BOX 3
            Enter address 2: SOMEWHERE FL 12345

    OUTPUT: DELIVERY POINT BAR CODE = 123450003039

!   !!  ! !  !!  !  ! ! ! !!   !!   !!     !! !!     !! ! !  !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

3.10 For centuries people have been intrigued by magic squares, with the earliest appearance about 2200 B.C. in China. A magic square of order 3 is an array of numbers having 3 rows and 3 columns such that the sum of every row, column, and diagonal equals the same number.

Write a program to display the unique magic square, given the sequence of numbers to use as input and the positions of two of those numbers (one in a corner and one in a middle position on a side). The first set of input will consist of the smallest number to use and the incremental difference of each succeeding number in the array. The second set of input will consist of two of those numbers, each followed by the row and column position it is to appear in the magic square. Every number in the array will be a positive integer less than 100 and each element of the array must be displayed right justified within a three column field. After skipping one line, display the magic number for which every column, row, and diagonal sums. Examples:

```
 INPUT: Enter first number: 2
        Enter increment: 3

        Enter number: 20
        Enter row, col: 1, 2

        Enter number: 23
        Enter row, col: 3, 1

OUTPUT:  17 20  5
          2 14 26
         23  8 11

        MAGIC NUMBER = 42


 INPUT: Enter first number: 25
        Enter increment: 5

        Enter number: 30
        Enter row, col: 3, 3

        Enter number: 65
        Enter row, col: 3, 2

OUTPUT:  60 25 50
         35 45 55
         40 65 30

        MAGIC NUMBER = 135
```