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 = 3.1415926535897932384626433832795028841971693993751058209
        74944592307816406286208998628034825342117067982148086 51
```

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: 4188.790204786390984616857844372670512262892532500141
            0946332594564104218750482786648373797671228227573095


     INPUT: Enter N: 90
            Enter radius: 100

    OUTPUT: 4188790.204786390984616857844372670512262892532500141
            094633259456410421875048278664837379767122822


     INPUT: Enter N: 85
            Enter radius: 55

    OUTPUT: 696909.970321335800065629723857503056477738745094710 9
            7461960854206028393946115736286231905 87
```

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
```