FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95

1.1 Write a program to display the following lines, each beginning at the left most column of the screen:

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF

1.2 Comments are used to explain sections of programming code. A comment statement is implemented a little differently in BASIC, Pascal, C, and C++. In BASIC, comments usually begin with an apostrophe (') but may also begin with the characters REM (short for REMARK). In Pascal, comments are usually delimited by curly braces ({ and }) but can be delimited by a parenthesis-asterisk and an asterisk-parenthesis { (* and *) }. In C, comments are only delimited by a slash-asterisk and an asterisk-slash (/* and */). In C++, comments usually begin with a double slash (//) but can be delimited by the same characters that the language C uses.

Write a program to accept as input a generic comment and then display it in the format commonly used by each of the languages mentioned. Example:

INPUT: Enter comment: **THIS PROGRAM WILL GENERATE COMMENTS**

OUTPUT: **BASIC: ' THIS PROGRAM WILL GENERATE COMMENTS**
PASCAL: { THIS PROGRAM WILL GENERATE COMMENTS }
C: /* THIS PROGRAM WILL GENERATE COMMENTS */
C++: // THIS PROGRAM WILL GENERATE COMMENTS

1.3 The C/C++ language includes two unary operators that are not found in other programming languages: the increment (++) and the decrement (--) operators. Increment adds one to a variable, and the decrement subtracts one from a variable; Thus, the language C++ is known as an incremental improvement over the language C.

Write a program to accept as input an integer N along with either the increment or decrement operator, and then display the new value for N: If N=-3 then N++ makes N=-2, and N-- makes N=-4. Examples:

INPUT: Enter N: **5**
Enter operator: **++**
OUTPUT: **6**

INPUT: Enter N: **20**
Enter operator: **--**
OUTPUT: **19**

1.4 Rounding off to three decimal places usually uses the rule that if the ten thousandth's digit is 5 or larger then round up, and if it is 4 or smaller then round down. The number 5 is called the break point. Write a program to accept as input a break point as any digit from 1 to 9 and accept as input a decimal number less than 10, and then display the number rounded to three decimal places. Examples:

```
 INPUT: Enter break point: 7      INPUT: Enter break point: 7
        Enter number: 1.3766             Enter number: 9.47979
OUTPUT: 1.376                    OUTPUT: 9.480
```

1.5 Time Sharing Option/Extensions (TSO/E) and Interactive System Productivity Facility (ISPF/PDF) are very useful for accessing a mainframe computer. REXX and CLIST are the two interpretative programming languages that can issue TSO and ISPF/PDF commands. REXX (REstructured eXtended eXecutor) and CLIST (Command LIST) code can be executed in the foreground within the MVS operating system. Since both languages are interpretative, each line is interpreted, and then executed, one line at a time, starting with the first line. The programs can be executed without being compiled and link-edited. BASIC is also commonly used as an interpretative programming language. Most other languages must be compiled and link-edited to create executable program modules.

CLIST is basically a command processor with limited programming functionality. REXX is a full application development programming language. REXX's structure and syntax closely resembles the language Pascal and to a lesser degree, PL/I. PL/I utilizes features from both COBOL (a business language) and FORTRAN (a scientific/mathematical language). REXX became the standard procedure language for IBM's System Application Architecture (SAA) in 1987, which means REXX is implemented across IBM's product line and used under several operating system environments. Starting with TSO/E Version 2, SYSPROC libraries can hold both CLIST and REXX programs, but the operating system needs to know what kind of program to execute. Therefore all REXX code stored in SYSPROC must start with a comment line that contains the word REXX on the first line.

Write a program to accept as input the first line of code as a comment and to display whether the operating system would interpret the program as a CLIST or a REXX. All comments begin with /* and end with */. If the four contiguous characters, REXX, appear somewhere in the first line of the comment then the operating system would interpret the code as a REXX program, otherwise it is seen as a CLIST program. Examples:

```
 INPUT: Enter comment: /* RESTRUCTURED EXTENDED EXECUTOR */
OUTPUT: CLIST

 INPUT: Enter comment: /* MY FIRST REXX-PROGRAM */
OUTPUT: REXX
```

1.6 In order to do well in a computer contest such as the FHSCC'95, a team must be quick in writing small programs. Any one of the following languages may be used in the contest: BASIC, Pascal, C, or C++. Beginners All-Purpose Symbolic Instruction Code (BASIC) was developed in the 1960's at Dartmouth College and was originally used on mainframes before becoming the most widely used programming language for microcomputers. BASIC is very easy to learn and the new versions contain powerful programming statements. Niklaus Wirth developed a new language in the early 1970's and named it after the 17th century mathematician, Blaise Pascal. Pascal is a highly structured and easy-to-maintain programming language that allows programmers to produce efficient programs. Dennis Ritchie at the Bell Telephone Laboratories developed a new language in order to design their UNIX operating system in 1972: C. Although C uses more special operators and symbols than most other languages (making it cryptic), C is the most popular professional programming language for microcomputers, enabling programmers to produce highly efficient code. AT&T's Bell Laboratories created the first C++ language in the 1980's to improve the way C works. C++ is an efficient language that has better C commands and the capability of using object-oriented programming (OOP).

Although BASIC, Pascal, and C/C++ are all good programming languages to use in this contest, programming in BASIC tends to be quicker to code since most variables do not need to be declared nor initialized. Pascal and C/C++ require that all variables be defined before they are used, whereas BASIC does not require a variable to be defined before it is used and automatically initializes all numeric variables to 0 and all strings to null. For example, to write the BASIC code "SUM = SUM + I + J" equivalently in either Pascal or C/C++ requires an additional appearance of the three variables by defining them before they are used (i.e. "var SUM, I, J: real;", or "float SUM, I, J;", respectively). An additional statement is required in Pascal to initialize SUM to zero (i.e. "SUM = 0;"), whereas C/C++ can initialize at the same time a variable is defined (i.e. "float SUM=0, I, J;"). Moreover, C/C++ can initialize variables "I" and "J" to a non-zero number at the same time they are defined, whereas both BASIC and Pascal must have a separate statement that assigns "I" and "J" to a non-zero number.

Write a program to accept as input the number of numeric variables used in a program and the number of those variables that need to be initialized and of those the number that need to be initialized specifically to zero, and then display the least number of times the variables must appear in declarations or statements before they may be used in a program for each of the three languages.

```
 INPUT: Enter number of variables: 6
        Enter number initialized: 4
        Enter number initialized to 0: 3
OUTPUT: BASIC = 1
        PASCAL = 10
        C/C++ = 6
```

1.7 Frank is called a "toolie" in the Configuration Management group on the CBSS project because he develops tools (programs using REXX) to assist the CM technicians in their daily tasks on the mainframe computer. Frank's programs need to read in files (data set names) and assimilate the last part of the data set name. A "qualified data set" name has all the information necessary to locate the data set via the system catalog and consists of two or more unqualified data set names connected by periods. These unqualified data set names (or qualifiers) consist of one to eight characters, the first being alphabetic or national (@, $, #) and the remaining characters must be alphameric, national, or a hyphen. The qualified data set names cannot be longer than 44 characters including the periods. The high-level qualifier is the first qualifier (or unqualified name) in the data set name.

Write a program to parse and display the last qualifier on a qualified data set name that is given as input. Examples:

```
 INPUT: Enter data set name: DT10005.REXX.EXEC
OUTPUT: EXEC

 INPUT: Enter data set name: G001246.CBSFCONV.SPUFI.IN.CBST012
OUTPUT: CBST012
```

1.8 Write a program to first accept as input a positive integer N less than 10. Next, the program is to accept as input N real numbers between -9999.9999 and 9999.9999, inclusive, and then display these N real numbers in reverse order on the screen, exactly as they were input. Example:

```
 INPUT: Enter N: 5
        Enter #: 1.23
        Enter #: -123.40
        Enter #: 999.9999
        Enter #: 0.0
        Enter #: -1234.4567

OUTPUT: -1234.4567
        0.0
        999.9999
        -123.40
        1.23
```

1.9 Write a program to display a large 'X' on the screen made up of letter X's. The program is to accept as input an odd number (between 3 and 15, inclusive) representing the number of X's to be displayed on each main diagonal. The top-left and bottom-left 'X' must appear in column 1 on the screen. Example:

INPUT: Enter number of X's: **7**

OUTPUT: (Screen clears and the following appears)

```
X     X
 X   X
  X X
   X
  X X
 X   X
X     X
```

1.10 GTE is an environmentally conscious corporate citizen. One of GTE's most environmentally impacting areas is the use of paper for phone bills. Considering this, GTE has decided to print bills on both sides of the paper (duplex) and save the equivalent of 3,000 80-foot trees per year. Printing on both sides reduces the number of pages in each bill by approximately 30 percent. In addition, this saves GTE a million dollars a month in the cost of forms and reduced postage charges. Assume that there are only six bill pages (front and back) in the first ounce of a bill due to the weight of the return envelope and an average of two inserts. Assume that each ounce after the first can have 9 pages (front and back). The long distance carriers have required that their bill pages not start on the back of any other carriers pages and that no other carrier appear on the back of their pages. GTE pays 23 1/3 cents per ounce for postage. Fractional ounces are paid using the next whole ounce. Tim has worked hard on developing this technology and would like to know how much GTE is saving so that management can adjust the postage budgets accordingly.

Write a program to accept as input the number of printed sides in a bill, and of those, the number of sides that will have a blank back side, and then determine the postage savings for that bill (in the format ###.## CENTS), compared to the cost of postage for a bill where all pages are printed on one side only. For example, if 50 sides are printed and 7 of those are single sided then (50-7)/2= 22 pages are double sided with duplex printing (total of 29 pages) as opposed to having 50 pages with single sided printing. Examples:

INPUT: Enter # of printed sides: **50**
Enter # of single sided pages: **7**
OUTPUT: **46.67 CENTS SAVED**

INPUT: Enter # of printed sides: **62**
Enter # of single sided pages: **10**
OUTPUT: **70.00 CENTS SAVED**

2.1 Write a program to find that integral solution of (X,Y) for AX+BY=C for which X is as small a positive integer as possible and A, B, and C are input as integers between -100 and 100, inclusive. Examples:

```
 INPUT: Enter A, B, C: 13, 21, 1
OUTPUT: (13,-8)

 INPUT: Enter A, B, C: 17, -19, 1
OUTPUT: (9,8)
```

2.2 Write a program to verify a "part number" by validating the check digit appearing in the units position. The computation of the check digit involves multiplying by 2 every other digit of the original number, starting with the first, and adding these values and the remaining digits of the number together. (Do not consider the right-most digit as a part of the number.) The units digit of the result obtained is then subtracted from 9 to obtain the check digit, which was not used in the computation. Using the part number in the first example below, 126547 becomes (1*2 + 6*2 + 4*2) + 2 + 5 = 29, ==> 9 - 9 = 0, and 0 does not match 7. The program is to accept as input a string of at most 20 digits, and display whether the part number is OKAY or in ERROR. If it is in ERROR, then display the correct check digit. Examples:

```
 INPUT: Enter part number: 126547
OUTPUT: ERROR -- CHECK DIGIT SHOULD BE 0

 INPUT: Enter part number: 1265400
OUTPUT: OKAY
```

2.3 Since computer education is the future of this nation, an imaginary millionaire's club would like to reward the efforts of the winners of the computer contest with 13 million dollars. Each of the winning teams will be awarded one of the following amounts: $1, $13, $169, $2197, $28561, $371293, and $4826809 (each a power of 13). If each prize (that is awarded) is given at most 9 times and the sum of all the awards total 13 million dollars, then write a program to determine how many of each prize will be awarded to the computer teams. Display the answer in the format below, where the symbol {#} represents a digit from 0 to 9:

```
OUTPUT: $1 = #
        $13 = #
        $169 = #
        $2197 = #
        $28561 = #
        $371293 = #
        $4826809 = #
```

2.4 Directory Assistance operators can be very beneficial to a person who does not know the entire phone number that they would like to call. A customer in the 813 area code may make up to three local Directory Assistance Calls (DAC) during each monthly billing period at no cost; Thereafter, each DAC in the local area costs 25 cents each. DAC's made within the 813 area code that are considered long distance are charged 25 cents per call. DAC's made to other area codes within Florida (i.e. 305, 407, 904) are charged 40 cents per call. DAC's made to places beyond Florida within the U.S. are charged 65 cents per call. International DAC's cost $3.00.

Write a program to first accept as input the number of DAC's made during the monthly billing period for a customer whose phone is in the 813 area code. The program is to then accept as input each DAC number and then display the cost associated with all these calls in the format ##.## DOLLARS. Each input will consist of at most 11 consecutive digits for a DAC. The DAC variations are shown below:

```
1411 ..................... Local Directory Assistance
1 + 813 + 555-1212  ....... Numbers within area code
1 + area code + 555-1212 .. Numbers outside area code
00 ....................... International calls
```

Example:

```
INPUT: Enter number of DAC's: 9
       Enter DAC: 1411
       Enter DAC: 1411
       Enter DAC: 18135551212
       Enter DAC: 00
       Enter DAC: 14075551212
       Enter DAC: 12025551212
       Enter DAC: 1411
       Enter DAC: 1411
       Enter DAC: 1411

OUTPUT:  4.80 DOLLARS
```

2.5 The new book entitled: FLORIDA HIGH SCHOOLS COMPUTING COMPETITION: PROBLEMS, JUDGING CRITERIA, BASIC SOLUTIONS, PASCAL SOLUTIONS: 1985 - 1994, by Douglas E. Woolley, contains 300 intriguing programming contest items and solutions. This 778 page book is a tool for enhancing computer programming skills and a preparation guide for those competing in contests such as this. The book is divided into four parts: Problems, Judging Criteria, BASIC solutions, and Pascal solutions. Each part is divided into 10 chapters corresponding to the years 1985 to 1994.

Write a program to display the heading of a page of the book given the page number as input. Assume that each chapter within a part has the same number of pages and that all pages are numbered consecutively and that the number of pages in each part is 180, 140, 200, and 260, respectively. Every even numbered page has in its heading the page number followed by 2 spaces and the title of the book: FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994. Every odd numbered page has in its heading the abbreviated title (FHSCC) followed by a space, followed by an apostrophe and the last two digits of the year of the contest and a space, followed by one of the four parts of the book: PROBLEMS, JUDGING CRITERIA, BASIC SOLUTIONS, or PASCAL SOLUTIONS; followed by two spaces and the page number. Examples:

INPUT: Enter page number: **8**
OUTPUT: **8 FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994**

INPUT: Enter page number: **51**
OUTPUT: **FHSCC '87 PROBLEMS 51**

INPUT: Enter page number: **181**
OUTPUT: **FHSCC '85 JUDGING CRITERIA 181**

INPUT: Enter page number: **755**
OUTPUT: **FHSCC '94 PASCAL SOLUTIONS 755**

INPUT: Enter page number: **444**
OUTPUT: **444 FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994**

2.6 The Internal Revenue Service (IRS) has compiled a table of "Estimated Preparation Time" to complete and file Form 1040 and its schedules:

Form	Recordkeeping	Learning about the law/form	Preparing the form	Copying, assembling, and sending form to IRS
Form 1040	3 hr., 8 min.	2 hr., 53 min.	4 hr., 41 min.	53 min.
Sch. A	2 hr., 32 min.	26 min.	1 hr., 10 min.	27 min.
Sch. B	33 min.	8 min.	17 min.	20 min.
Sch. C	6 hr., 26 min.	1 hr., 10 min.	2 hr., 5 min.	35 min.
Sch. D	51 min.	42 min.	1 hr., 1 min.	41 min.
Sch. E	2 hr., 52 min.	1 hr., 7 min.	1 hr., 16 min.	35 min.

Write a program to enter at most 6 unique forms and display the total ESTIMATED PREPARATION TIME to complete all stages of the forms designated. Valid input will consist of: 1040, A, B, C, D, or E; Input is terminated by an invalid entry. Output will be displayed with the minutes between 0 and 59 inclusive. Examples:

```
 INPUT: Enter form: D
        Enter form: 1040
        Enter form: NO

OUTPUT: 14 HR., 50 MIN.


 INPUT: Enter form: B
        Enter form: F

OUTPUT: 1 HR., 18 MIN.
```

2.7 At GTE there are many investment incentives for employees, such as the 401K investment plan and the guaranteed stock returns.

The 401K is a plan where an employee can contribute up to 16% of his/her income into investment funds. The company will match each dollar with 75 cents, of the first 6% contributed, with a Company-Matching Contribution, credited to the employee's account at the end of the year. These combined funds have returned a yearly interest rate, ranging from 6% to 29%, that is added to the employee's account.

Under the terms of the stock purchase plan, an employee can purchase one share of common stock for each full $100 of their annual basic rate of pay up to a maximum of 750 shares. The company sells each share to the employee at a guaranteed 85% of the "Average Market Price." If stock prices are higher at the end of the year than at the beginning, then the employee could earn more than 15% but never less.

Write a program that will allow an employee to see how much he can profit by investing. Input will first consist of the yearly salary and the percent of the 401K that will be contributed. Next, the program is to display the number of shares the employee can purchase, and then accept as input the number of shares that are bought, followed by the closing market price of a share (which will be greater than the starting value). Assume that stock prices (or "Average Market Price") start at $32.00 per share at the beginning of the year (thus employees purchase a share at $32.00 * 0.85), and 14% is the return on the combined employee/company contributions to the 401K. Output must consist of the company contribution, the 401K return, the gain in stock, and the total of these three gains, all in the form #####.##. Examples:

```
 INPUT: Enter salary: 32080
        Enter 401K %: 16
OUTPUT: YOU CAN PURCHASE UP TO 320 SHARES
 INPUT: Enter number of shares: 320
        Enter end of year price: 35.00
OUTPUT: COMPANY CONTRIBUTION:  1443.60
              401K RETURN:   920.70
               STOCK GAIN:  2496.00
               TOTAL GAIN:  4860.30


 INPUT: Enter salary: 54321
        Enter 401K %: 4
OUTPUT: YOU CAN PURCHASE UP TO 543 SHARES
 INPUT: Enter number of shares: 100
        Enter end of year price: 33.25
OUTPUT: COMPANY CONTRIBUTION:  1629.63
              401K RETURN:   532.35
               STOCK GAIN:   605.00
               TOTAL GAIN:  2766.98
```

2.8 Write a program to replicate the following pattern on the screen, given as input the number of spiral loops to display (less than 6) and the letter to be used in the first spiral. Each succeeding spiral will use the next letter in the alphabet (except Z is followed by A). After accepting input, the screen clears and the first character of the spiral is centered on the screen. Example:

INPUT: Enter number of spiral loops: **4**
Enter first letter: **Y**

OUTPUT: (Screen clears and the following is centered)

```
B
B BBBBBBBBBBBBBB
B A            B
B A AAAAAAAAAA B
B A Z        A B
B A Z ZZZZZZ A B
B A Z Y    Z A B
B A Z Y YYY Z A B
B A Z Y   Y Z A B
B A Z YYYYY Z A B
B A Z       Z A B
B A ZZZZZZZZZ A B
B A           A B
B AAAAAAAAAAAAA B
B               B
BBBBBBBBBBBBBBBB
```

2.9 Write a program to display all possible moves for the Queen on an empty chess board. The program is to first accept as input the coordinates of the Queen (column A-H followed by row 1-8), and then clears the screen and displays the board layout in the upper left corner of the screen. A Queen may move horizontally, vertically, or diagonally. The letter 'Q' marks the position of the Queen and asterisks mark the possible moves for the Queen. Example:

INPUT: Enter column and row: **C4**

OUTPUT: (Screen clears and the following appears)

```
8     *       *
7     *     *
6 *   *   *
5   * * *
4 * * Q * * * * *
3   * * *
2 *   *   *
1     *     *
  A B C D E F G H
```

2.10 During a pre-election poll, information was collected concerning: sex, age, race, income, and party to vote for. Each set of these categories will continue to be input until an 'E' is entered for sex. Valid inputs for sex are: M for Male, F for Female, or E to End; for race: W for White, or O for Other; and for party: D for Democratic, or R for Republican. Write a program to tabulate the data collected and generate a report showing percentages of each category among the Democratic and Republican parties as shown below. The column headings DEMOCRATIC and REPUBLICAN begin in columns 33 and 45 respectively. Percentages are displayed in the format ###.#. Example:

```
INPUT: Enter sex: M
       Enter age: 23
       Enter race: O
       Enter income: 19000
       Enter party: R

       Enter sex: F
       Enter age: 67
       Enter race: W
       Enter income: 34000
       Enter party: R

       Enter sex: M
       Enter age: 51
       Enter race: W
       Enter income: 56000
       Enter party: D

       Enter sex: E
```

OUTPUT:

	DEMOCRATIC	REPUBLICAN
MALE	33.3	33.3
FEMALE	0.0	33.3
50 AND BELOW	0.0	33.3
OVER 50	33.3	33.3
WHITE	33.3	33.3
OTHERS	0.0	33.3
ABOVE $25000	33.3	33.3
$25000 AND BELOW	0.0	33.3
WHITE MALE OVER 50 AND ABOVE $25000	33.3	0.0
OTHER	0.0	66.7

3.1 As the first quarter of the year approaches, many people are working on their tax return. Write a program to determine how much money an individual tax payer will either pay to the IRS or receive back from the IRS.

The program is to first accept as input the adjusted gross income of a single person and the amount of itemized deductions. If the deductions are greater than the standard deduction of $3,800, then subtract the itemized amount from the adjusted gross income; otherwise subtract $3,800 from the adjusted gross income. Subtract an additional $2,450 (for one claimed exemption) to produce the taxable income.

Each year the IRS produces a tax table corresponding to taxable income less than $100,000. For incomes that exceed $100,000, a tax rate schedule is used instead. Even though the tax rate schedule is not used for income less than $100,000, this formula will be used in your program for all income levels and will produce amounts within $8 of an actual tax table look-up for incomes less than $100,000. Tax is computed as follows:

```
15%   of the first $22,750 plus
28%   of the amount over  $22,750 up to  $55,100 plus
31%   of the amount over  $55,100 up to $115,000 plus
36%   of the amount over $115,000 up to $250,000 plus
39.6% of the amount over $250,000
```

The program is also to accept as input the amount of federal income tax withheld from the person. If this amount is less than the computed tax, then the difference is owed to the IRS; otherwise the IRS owes the difference. Display the amount owed in the format: ######.## DOLLARS. Examples:

```
 INPUT: Enter adjusted gross income: 32140.65
        Enter itemized deductions: 4758.00
        Enter federal income tax withheld: 4062.00

OUTPUT:     38.36 DOLLARS WILL BE REFUNDED TO YOU


 INPUT: Enter adjusted gross income: 306250.00
        Enter itemized deductions: 3456.00
        Enter federal income tax withheld: 11222.00

OUTPUT:  88217.50 DOLLARS YOU OWE
```

3.2 GTE has become the largest U.S.-based local exchange carrier, with more than 22 million access lines worldwide. The GTE phone company uses a complex computer application called CBSS to bill its valued customers. In a simplified fashion, GTE charges customers a long distance rate determined by the length of a phone call in minutes, the time of day the call was placed, and to where the call was placed. Assuming that the rate chart below is in effect, write a program to produce a simplified phone bill for a customer given a series of phone calls input. Each phone call consists of the length in MINutes, and the time in the form: HH:MM AM DAY
where AM could also be PM, and DAY is the first 3 letters of the name (i.e. MON, TUE, WED, THU, FRI, SAT, SUN). The last call recorded is indicated by entering a 0 for the next prompt of MIN. If the customer's bill is over $20, then give a 20% discount, otherwise give a 0% discount. Display the times without the leading zero as shown below, and compute the total charges, the discount, and the charges minus the discount. Assume that the phone bill is for BOB SMITH who calls from his home at 813-555-1234 and always makes calls to the same long distance number. Note: 11am is followed by 12pm, then 1pm; 11pm is followed by 12am, then 1am. The rates are as follows (the first rate is for the first minute, the second rate is for all subsequent minutes):

Weekday Rates (Mon - Fri)		Weekend Rates (11pm Fri - 7:59am Mon)	
8am - 4:59pm	.28 / .21		.14 / .11
5pm - 10:59pm	.21 / .16	(except 5pm Sun - 10:59pm Sun)	
11pm - 7:59am	.14 / .11		.21 / .16

Example:

```
 INPUT: Enter MIN: 1
        Enter time: 07:56 AM  MON
        Enter MIN: 25
        Enter time: 12:01 PM  THU
        Enter MIN: 35
        Enter time: 03:15 PM  SAT
        Enter MIN: 84
        Enter time: 11:59 AM  FRI
        Enter MIN: 20
        Enter time: 10:09 AM  WED
        Enter MIN: 0

OUTPUT: BOB SMITH  (813) 555-1234

          TIME OF DAY  MIN.  CHARGE
         7:56 AM  MON    1    0.14
        12:01 PM  THU   25    5.32
         3:15 PM  SAT   35    3.88
        11:59 AM  FRI   84   17.71
        10:09 AM  WED   20    4.27

        TOTAL CHARGES        31.32
        DISCOUNT              6.26
        CHARGES - DISCOUNT   25.06
```

3.3 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

3.4 Write a program that will accept up to 8 distinct letters in a string and output a list of all possible subsets of the list. Each subset will be listed alphabetically within the subset and in ascending order amongst the other subsets. The output must have as many complete subsets on a 50-character line as possible, with one space separating each subset. On the line after the last set of subsets, the total number of subsets must be displayed. Example:

INPUT: Enter letters: **CABFE**
OUTPUT: **{} {A} {AB} {ABC} {ABCE} {ABCEF} {ABCF} {ABE}**
{ABEF} {ABF} {AC} {ACE} {ACEF} {ACF} {AE} {AEF}
{AF} {B} {BC} {BCE} {BCEF} {BCF} {BE} {BEF} {BF}
{C} {CE} {CEF} {CF} {E} {EF} {F}
TOTAL SUBSETS = 32

INPUT: Enter letters: **ZYX**
OUTPUT: **{} {X} {XY} {XYZ} {XZ} {Y} {YZ} {Z}**
TOTAL SUBSETS = 8

3.5 Write a program for Mr. Gauss to accurately and efficiently compute the sum of the integers from 1 to N, where N is input as a positive integer having less than 40 digits. Examples:

```
 INPUT: Enter number N: 10000000000000000000000000000000
OUTPUT:
50000000000000000000000000000005000000000000000000000000000000
```

```
 INPUT: Enter number N: 99999999999999999999999999999999
OUTPUT:
49999999999999999999999999999999950000000000000000000000000000000000
```

3.6 Write a program to input several lines of BASIC code and display the final values of all the variables used. All statements are executed in the order input and are of the form:

variable = <variable/constant> [<operator> <variable/constant>]

- variable is any single letter
- constant is any single digit
- operator is +, -, *, or /.

The last line of the program will be indicated by 'END'. All variables used on the right side of the equal sign {=} will have been previously assigned a value. All variables are to be displayed in the order that they are used in the program and all the values displayed will be integers. Examples:

```
 INPUT: Enter line: A=5
        Enter line: B=9
        Enter line: A=B+7
        Enter line: B=A-B
        Enter line: END

OUTPUT: A=16
        B=7
```

```
 INPUT: Enter line: J=2
        Enter line: E=J*3
        Enter line: S=E
        Enter line: U=7*7
        Enter line: S=J-5
        Enter line: J=2+E
        Enter line: E=E/2
        Enter line: END

OUTPUT: J=8
        E=3
        S=-3
        U=49
```

3.7 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
### + ### + ### = ####
### + ### + ### = ####
### + ### + ### = ####
```

3.8 Write a program to clear the screen and display a runner's digital stop-watch time in block numbers given the minutes and seconds as input. The time must increment by one second approximately every second: No more than 15 seconds and no less than 7 seconds are to be displayed every 10 actual seconds. Program terminates upon pressing any key. All times are to be displayed in the upper-left corner of the screen in block numbers 4 asterisks wide and 5 asterisks high:

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

Example:

INPUT: Enter MM:SS: **09:58**

OUTPUT: (Screen is cleared and the time is displayed in the upper-left corner of screen)

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

(approximately 1 second later the following appears)

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

(approximately 1 second later the following appears)

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

(approximately 1 second later the following appears)

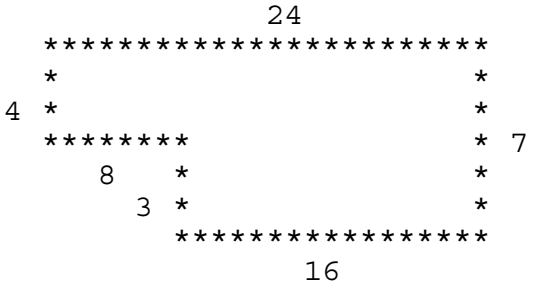
```
   *  ****     ****     *
   *  *  *  *  *  *     *
   *  *  *     *  *     *
   *  *  *  *  *  *     *
   *  ****     ****     *
```

INPUT: (press any key)

OUTPUT: (program terminates)

3.9 GTE Data Services was incorporated on Oct. 25, 1967 and has recently restructured its four regional Information Processing Centers (IPCs) into three Information Control Centers (ICCs) in Tampa, Florida; Sacramento, California; and Fort Wayne, Indiana. Each of the buildings located in these areas have many different rooms and work 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 Jim is the Distribution Coordinator for GTEDS CBSS Project. One aspect of his job is to assign which week of the year to create a new version of a library of data sets (files). The following is his base criteria:

- Each Version of a library spends 12 weeks in a test area and is named R1VvvL01, where vv is the Version number;
- Immediately following this test phase, the library is moved to Production for 6 weeks;
- 1 week before a library is moved to Production, a new Pre-Production test library is created and is functional for 6 weeks; This library is called R1VvvL88;
- 6 weeks after a Version enters a test area, the next Version of the library goes to the other test area; This Version follows the same time frames as listed above and is named similar to the previous Version, except that this Version is one greater in number;
- There are 2 test areas, and they alternate Versions of the library; all even Versions are in Test 1; all odd in Test 2.

Write a program to display the time relationships of these library versions by a horizontal graph. Input will consist of (a) a version number; (b) the week number that it goes to a test area; (c) the first week and the number of weeks to display on the graph (each less than 50). The program is to clear the screen and then display each week number vertically, starting in column 10. The versions are to be displayed in the order that they are created, each beginning in column 1. The program must show the time a version is in a test area by displaying a 1 or a 2. Display the weeks that a version is in Production with a P. Display the weeks a version has a Pre-Production test area with an asterisk. Example:

```
 INPUT: Enter version #: 36
        Enter first week in test: 2
        Enter first week to display, # of weeks: 4, 34
OUTPUT: (Screen clears and the following displays)
                 00000011111111112222222233333333
                 45678901234567890123456789012345678

        R1V34L01 PPPP
        R1V35L01 2222PPPPPP
        R1V34L88 ***
        R1V36L01 1111111111PPPPPP
        R1V35L88    ******
        R1V37L01     222222222222PPPPPP
        R1V36L88          ******
        R1V38L01           111111111111PPPPPP
        R1V37L88                ******
        R1V39L01                 222222222222PPPPPP
        R1V38L88                      ******
        R1V40L01                       111111111111
        R1V39L88                            ******
        R1V41L01                             222222
        R1V40L88                                  *
```