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: 12345678901234567890З
            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:

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

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