# FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '96

**1.1** FHSCC is an abbreviation for Florida High Schools Computing Competition. Write a program to accept as input a four-digit year between 1980 and 1996, inclusive, and append the last two digits of the year to the phrase: **FHSCC '**. Examples:

```
 INPUT: Enter year: 1996
OUTPUT: FHSCC '96

 INPUT: Enter year: 1984
OUTPUT: FHSCC '84
```

**1.2** Write a program to tally the number of frequent flier miles that Doug earns if he flies to and from Caracas, Venezuela **X** times and stays at the Hilton each time and pays a total of **Y** dollars in phone calls. The distance of the flight is 1300 miles one-way. Each "stay" at the Hilton earns 500 miles, and each dollar spent on phone calls earns 5 miles. Total will be less than 32767. Examples:

```
 INPUT: Enter X: 3             INPUT: Enter X: 2
        Enter Y: 10                   Enter Y: 100
OUTPUT: 9350                  OUTPUT: 6700
```

**1.3** Write a program to print the middle letter or letters of a given word. If the word has an even number of letters, then there will be two middle letters. If the word has an odd number of letters, then there will be one middle letter. Examples:

```
 INPUT: Enter word: DOUG
OUTPUT: OU

 INPUT: Enter word: WOOLLEY
OUTPUT: L
```

**1.4** Write a program to accept the two end coordinates (X,Y) of one of the diagonals of a rectangle in the Cartesian plane whose sides are parallel to the X or Y-axis. The program must then display the area and perimeter of the rectangle. Examples:

```
 INPUT: Enter coordinate 1: -1, 2
        Enter coordinate 2: 4, -2
OUTPUT: AREA = 20
        PERIMETER = 18

 INPUT: Enter coordinate 1: 3, 1
        Enter coordinate 2: 0, 0
OUTPUT: AREA = 3
        PERIMETER = 8
```

**1.5** Write a program to code-break a given encrypted secret message made up of alphabetic characters and spaces. The decoder must translate the letters ABCDEFGHIJKLMNOPQRSTUVWXYZ into the corresponding letters ZYXWVUTSRQPONMLKJIHGFEDCBA, respectively. A space is decoded into a space. The encryption will not contain more than 40 characters. Example:

```
 INPUT: Enter encryption: UOLIRWZ SRTS HXSLLO
OUTPUT: FLORIDA HIGH SCHOOL

 INPUT: Enter encryption: XLNKFGVI XLMGVHG
OUTPUT: COMPUTER CONTEST
```

**1.6** A nice hotel in Caracas, Venezuela has 26 floors above the ground floor. Write a program to accept as input the floors on which an elevator of this hotel stops consecutively, and determine the total number of floors that an elevator of this hotel touches and the number of unique floors that the elevator touches. The elevator always starts on the ground (floor 0) and ends on the ground (floor 0). Therefore, the elevator starts by touching the ground floor and ends touching the ground floor. In the first example below, the elevator touches floors 0,1,2,3,4,5 then floors 4,3, then floor 4, then floors 3,2,1,0 for a total of 13 floors. In the second example below, the elevator touches floors 0,1,2, then floors 3,4, then floors 3,2,1,0 for a total of 9 floors. Examples:

```
 INPUT: Enter floor: 5
        Enter floor: 3
        Enter floor: 4
        Enter floor: 0

OUTPUT: TOTAL FLOORS TOUCHED = 13
        UNIQUE FLOORS TOUCHED = 6


 INPUT: Enter floor: 2
        Enter floor: 4
        Enter floor: 0

OUTPUT: TOTAL FLOORS TOUCHED = 9
        UNIQUE FLOORS TOUCHED = 5


 INPUT: Enter floor: 20
        Enter floor: 5
        Enter floor: 24
        Enter floor: 10
        Enter floor: 0

OUTPUT: TOTAL FLOORS TOUCHED = 79
        UNIQUE FLOORS TOUCHED = 25
```

**1.7** Write a program to determine a person's ratios for buying a house and whether he qualifies for a mortgage if a mortgage company will not approve a loan with ratios over 33% / 38%. All amounts input are monthly tallies. To qualify for the loan, the first ratio must not exceed 33% and the second ratio must not exceed 38%. The first ratio computes the loan amount divided by the income. The second ratio computes the sum of the loan and other debts divided by the income. Display the first and second ratios in the format: RATIOS = ##.#% / ##.#%, where each ratio is rounded to the nearest tenth of a percent . Display if this person DOES QUALIFY or DOES NOT QUALIFY for the loan. Examples:

```
 INPUT: Enter amount of loan: 900
        Enter amount of debts: 200
        Enter amount of income: 2800

OUTPUT: RATIOS = 32.1% / 39.3%
        DOES NOT QUALIFY


 INPUT: Enter amount of loan: 1000
        Enter amount of debts: 170
        Enter amount of income: 3100

OUTPUT: RATIOS = 32.3% / 37.7%
        DOES QUALIFY
```

**1.8** Write a program to convert numbers 1-10 to the English or Spanish word for the number. The program will first prompt for either the letter 'E' for English or 'S' for Spanish. Next, the program will accept as input a number between 1 and 10, inclusive, and display the name of the number in the requested language.

English: ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE TEN
Spanish: UNO DOS TRES CUATRO CINCO SEIS SIETE OCHO NUEVE DIEZ

```
 INPUT: Enter E or S: E
        Enter number: 4

OUTPUT: FOUR


 INPUT: Enter E or S: S
        Enter number: 10

OUTPUT: DIEZ


 INPUT: Enter E or S: S
        Enter number: 5

OUTPUT: CINCO
```

**1.9** Write a program to form a cross with the word(s) input, given that the total number of characters input is an odd number and the word(s) are to intersect at the middle character. Example:

```
 INPUT: Enter word(s): THE CROSS
OUTPUT:     T
            H
            E

        THE CROSS
            R
            O
            S
            S
```

**1.10** Write a program to simulate the PRICE IS RIGHT game. Given an item's actual price, and unique price guesses from four contestants, determine which person comes the closest to the price without going over the actual cost. If every contestant goes over the price, then display the message "EVERYONE IS OVER". Examples:

```
 INPUT: Enter actual price: 425
        Enter guesses A, B, C, D: 300, 400, 500, 200
OUTPUT: PERSON B

 INPUT: Enter actual price: 399
        Enter guesses A, B, C, D: 300, 400, 500, 301
OUTPUT: PERSON D

 INPUT: Enter actual price: 299
        Enter guesses A, B, C, D: 300, 400, 500, 301
OUTPUT: EVERYONE IS OVER
```

**2.1** Write a program that will emulate random dart throws. The dart scores possible are 0,2,4,5,10,20,50 with the probability of hitting any score being the same as any other. The object of the game is to accumulate a score of at least 100 points. The program will print the score of each dart throw, separated by a comma, until the sum of the scores totals 100 points or more. The program must then print the number of throws that achieved the score, followed by the total score achieved. Sample RANDOM runs:

```
OUTPUT: 2,4,20,4,10,0,5,20,4,2,50
          11 THROWS ACHIEVED SCORE OF 121

OUTPUT: 50,20,10,5,10,2,5
          7 THROWS ACHIEVED SCORE OF 102
```

**2.2** Write a program to compress information and save space, given a string of data. Input will consist of a string of letters with one or more asterisks (representing spaces) between words. The program must display the string of data with multiple asterisks replaced by the number of asterisks being eliminated. If only one asterisk separates two words, the asterisk should not be replaced with the number 1 since space would not be conserved. Examples:

```
 INPUT: Enter string: WE*CONSERVE****SPACE**BY***COMPRESSION
OUTPUT: WE*CONSERVE4SPACE2BY3COMPRESSION

 INPUT: Enter string: THIS**SENTENCE*IS*****COMPRESSED
OUTPUT: THIS2SENTENCE*IS5COMPRESSED
```

**2.3** Set "A" has the property that the product of any two elements (numbers in the set) is 1 less than a perfect square. The numbers 1, 3, and 8 are elements of this set, i.e. 1x3=(4-1), 1x8=(9-1), 3x8=(25-1). Write a program to find two other whole numbers less than 1000 that are also elements of set "A". Display the solutions in numerical order in the format shown below, where # represents a digit. Example output format:

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

**2.4** Write an efficient program to display the least common multiple of a set of integers from 1 to N, inclusive, where N is at most 30. The least common multiple is a positive integer that is evenly divisible by every integer in the set and will contain at most 13 digits. Examples:

```
 INPUT: Enter N: 6              INPUT: Enter N: 30
OUTPUT: 60                     OUTPUT: 2329089562800
```

**2.5** Write a program to calculate a fractional value of three-letter words. The reciprocal value of a letter in the alphabet (A...Z) is defined as the reciprocal of the position of that letter in the alphabet (A=1/1, B=1/2, C=1/3 ... Z=1/26). The fractional value is the sum of the reciprocals of the value of each letter in that word. This value must be printed as a simplified fraction. In the first example below, CAB is 1/3 + 1/1 + 1/2 = 11/6. In the second example below, EAT is 1/5 + 1/1 + 1/20 = (4 + 20 + 1)/20 = 25/20 = 5/4. Examples:

```
 INPUT: Enter word: CAB
OUTPUT: 11/6

 INPUT: Enter word: EAT
OUTPUT: 5/4
```

**2.6** The Fibonacci sequence has the property that each number (beyond the first two) is the sum of the previous two numbers (1,1,2,3,5,8,...). The first two numbers in the sequence are 1 and 1. The third number is the sum of 1 and 1, or 2. The fourth number is the sum of 1 and 2, or 3. The fifth number is the sum of 2 and 3, or 5, etc.

Write a program to accept as input a number, N, less than 10, and display the Nth prime within the Fibonacci sequence. The first prime number in the sequence is the number 2. Examples:

```
 INPUT: Enter N: 3
OUTPUT: 5

 INPUT: Enter N: 9
OUTPUT: 514229
```

**2.7** GTE sorts its phone bills by postal code then by telephone number before the bills are printed. Write a program to accept as input a list of (at most 8) four-digit phone numbers followed by zip code, and display the order in which the phone numbers will be printed. Input will be terminated by the entry **0000, 00000**. All other entries will not have any leading zeros. Example:

```
 INPUT: Enter phone #, zip: 1796, 33647
        Enter phone #, zip: 1521, 33555
        Enter phone #, zip: 2001, 33647
        Enter phone #, zip: 1400, 33647
        Enter phone #, zip: 1621, 33555
        Enter phone #, zip: 0000, 00000

OUTPUT: 1521
        1621
        1400
        1796
        2001
```

**2.8** Write a program to display the findings of a statistical test on a set of random letters. Given a string of letters, display the number of runs of letters that are in the first half of the alphabet (A,B,C,D,E,F,G,H,I,J,K,L,M), and the number of runs of letters that are in the second half of the alphabet (N,O,P,Q,R,S,T, U,V,W,X,Y,Z). A run is a continuous group of elements in the same category. A string of FLANOMZUGODISGOODF consists of the runs: FLA, NO, M, ZU, G, O, DI, S, G, OO, DF. Examples:

```
 INPUT: Enter letters: FLANOMZUGODISGOODF
OUTPUT: RUNS IN 1ST HALF = 6
        RUNS IN 2ND HALF = 5

 INPUT: Enter letters: XPQJESUSISLORDQPY
OUTPUT: RUNS IN 1ST HALF = 4
        RUNS IN 2ND HALF = 5
```

**2.9** Write a program to reverse the order of letters in each word of a given string unless the word is a palindrome (a word that is spelled the same forward and backward). If the word is a palindrome, then replace each letter with a question mark (?). Each word is separated by a single space. Examples:

```
 INPUT: Enter string: HOW GOOD IT IS FOR REMER
OUTPUT: WOH DOOG TI SI ROF ?????

 INPUT: Enter string: OTTO CAME UP WITH THE WORD ROADAOR
OUTPUT: ???? EMAC PU HTIW EHT DROW ???????
```

**2.10** Write a program to determine the day of the week that a given date falls by using the following three tables and algorithm:

| MONTH NUMBERS | | |
|---|---|---|
| January | 1 | (if leap year then use 0) |
| February | 4 | (if leap year then use 3) |
| March | 4 | |
| April | 0 | |
| May | 2 | |
| June | 5 | |
| July | 0 | |
| August | 3 | |
| September | 6 | |
| October | 1 | |
| November | 4 | |
| December | 6 | |

| CENTURY NUMBERS | |
|---|---|
| 1753 to 1799 | add 4 |
| 1800 to 1899 | add 2 |
| 1900 to 1999 | add 0 |
| 2000 to 2099 | add 6 |
| 2100 to 2199 | add 4 |

| DAY NUMBERS | |
|---|---|
| Saturday | = 0 |
| Sunday | = 1 |
| Monday | = 2 |
| Tuesday | = 3 |
| Wednesday | = 4 |
| Thursday | = 5 |
| Friday | = 6 |

Sum the following five derived numbers:

1) The last two digits of the year.
2) The whole quotient of this number divided by 4.
3) The MONTH NUMBER associated with the input month.
4) The day of the month for the input date.
5) The CENTURY NUMBER associated with the input year.

Next, divide the sum of the five numbers above by 7 and compare the remainder with the DAY NUMBERS to obtain the corresponding day.

Input will be three numbers corresponding to the month, day, and year. Note that a leap year is divisible by 4, except for those years also divisible by 100; but, if the year is also divisible by 400 then it is still a leap year. In the first example below, the algorithm uses the input date of August 4, 1856, and divides the sum of (56 + 14 + 3 + 4 + 2) by 7, which equals 11 remainder 2. The number 2 corresponds to Monday, so August 4, 1856 was a Monday. Examples:

```
 INPUT: Enter month, day, year: 8, 4, 1856
OUTPUT: MONDAY

 INPUT: Enter month, day, year: 9, 27, 1990
OUTPUT: THURSDAY

 INPUT: Enter month, day, year: 2, 1, 1996
OUTPUT: THURSDAY
```

**3.1** Write a program to display the appearance of a 3-dimensional book with the two title lines centered vertically on the spine. The spine's height is to be 2 characters more than the longest title line. The book's width is to be 5 characters. In the process of centering the shorter title line, if there is an extra space it should succeed the title line. Title lines will not exceed 17 characters. The left side of the book must be in column 1 and the top of the book must be on row 1 of the screen. Examples:

```
     INPUT: Enter title 1: WHERE WE
            Enter title 2: STAND
    OUTPUT: (Screen clears, left side in column 1, top in row 1)
                /---/!
               /   / !
              /   /  !
             /   /   !
            !---!    !
            !  W!    !
            !S H!    !
            !T E!    !
            !A R!    !
            !N E!    !
            !D  !   /
            !  W!  /
            !  E! /
            !---!/


     INPUT: Enter title 1: EVIDENCE THAT
            Enter title 2: DEMANDS A VERDICT
    OUTPUT: (Screen clears, left side in column 1, top in row 1)
                /---/!
               /   / !
              /   /  !
             /   /   !
            !---!    !
            !D  !    !
            !E  !    !
            !M E!    !
            !A V!    !
            !N I!    !
            !D D!    !
            !S E!    !
            !  N!    !
            !A C!    !
            !  E!    !
            !V  !    !
            !E T!    !
            !R H!    !
            !D A!    !
            !I T!   /
            !C  !  /
            !T  ! /
            !---!/
```

**3.2** Write a program to produce a prime factors tree for a given number. As seen in the example below, the symbols {/} and {\} appear beneath the largest non-prime factors' left and right, respectively. The smallest prime factors on the bottom-left of each {/} are to be displayed in ascending order on each succeeding pair of lines. The larger factor on each line is the dividend of the number appearing above it and the prime on its left. The input number will have less than 10 prime factors and none of the factors will have more than four digits. The program must clear the screen and display the input number beginning in column 6. Examples:

```
     INPUT: Enter number: 100

    OUTPUT: (Screen is cleared, and the following appears)
                 100
                /   \
               2     50
                    /  \
                   2    25
                       /  \
                      5    5


     INPUT: Enter number: 1716

    OUTPUT: (Screen is cleared, and the following appears)
                 1716
                /    \
               2      858
                     /   \
                    2     429
                         /   \
                        3     143
                             /   \
                           11     13
```

**3.3** Write a program to simulate a "base four" calculator that accepts expressions involving +, -, and unsigned integers in base 4. Each integer input will be less than 6 digits long and the result must be displayed in base 4. No expression will contain more than 40 characters. Examples:

```
     INPUT: Enter base 4 expression: 1230+23-3210-123+10
    OUTPUT: -2010

     INPUT: Enter base 4 expression: 123-12-12+23-321+333
    OUTPUT: 200
```

**3.4** Write a program to calculate the daily amount of money that a contractor makes for working between a given time frame. Input will consist of the hourly rate of pay in dollars for contractors who work more than 50 percent of their time during normal working hours (7 AM - 5 PM). If at least 50 percent of the work is done outside of normal hours then the pay rate is to be increased by a "shift differential" of 10 percent to be applied for all the hours worked. Contractors will work less than 12 hours per day. The start and finish time will be input in the form HH:MM and followed by either AM or PM. Note: 11:59AM is followed by 12:00PM, and 11:59PM is followed by 12:00AM. Output must be of the format $DDD.CC with no leading zeros in dollars and have trailing zeros in cents. Examples:

```
 INPUT: Enter pay/hour: 9.05
        Enter start time: 08:00AM
        Enter finish time: 07:00PM
OUTPUT: $ 99.55

 INPUT: Enter pay/hour: 20.00
        Enter start time: 12:15PM
        Enter finish time: 09:45PM
OUTPUT: $209.00
```

**3.5** On a panel of 16 buttons (4 rows of 4 buttons), each button must be pressed once and in the correct order. The final button to be pressed is always marked 0F for Final. The number of moves and the direction is marked on each button. 1R means one move right; 2D means two moves down; 3U means three moves up; 1L means one move left.

Write a program to print the first button that you must press (and its position) that will lead you to press every other button and finish at the final button 0F. In the first example below, pressing the 3L button leads to 1D, 3R, 2U, 1U, 2L, 3D, 1R, 3U, 2L, 1D, 2R, 1L, 1D, 1R, 0F. Examples:

```
 INPUT: Enter row: 1D 3D 2L 2L
        Enter row: 2R 1D 1L 1U
        Enter row: 1D 1R 0F 3L
         Enter row: 3R 1R 3U 2U

OUTPUT: FIRST BUTTON = 3L
        AT ROW = 3, COL = 4


 INPUT: Enter row: 0F 3D 3D 2L
        Enter row: 2R 1D 1U 2L
        Enter row: 2U 1L 1R 1U
         Enter row: 2U 1L 1U 3U

OUTPUT: FIRST BUTTON = 3U
        AT ROW = 4, COL = 4
```

**3.6** A magic square is a matrix of distinct numbers with the same number of rows as columns, and the sum of the numbers in each row, column, and diagonal is equal to the same total (the magic number). There are a number of general methods for generating magic squares with an odd "order" (number of rows and columns). La Loubre invented the staircase method:

1) Start with a number in the top middle square.

2) The next number (incremented by a constant) is placed diagonally up and right in the next box of the array. If the number would be placed outside of the array, then the number is moved to another spot in the array according to these two rules: If the top row is exceeded, then it is placed in the bottom row; if the right-most column is exceeded, then it is placed in the first column.

3) If the square is already occupied while trying to place the number in the array, then the number is placed in the square that is immediately below the original number. If the bottom row is exceeded then the number is placed in the top row with the same column.

4) Continue to place numbers in the magic square by repeating steps 2 and 3 until all squares have been populated.

Write a program to display a magic square using the staircase algorithm, given as input an odd "order" for the magic square (at most 13), the first positive integer to use, and the positive integral increment between each successive number. In the magic square, each number is to be right justified within a four-character column. Begin the output by displaying the magic number (the sum of all the cells for a column, for a row, for a diagonal). Examples:

```
 INPUT: Enter order, first number, increment: 3, 2, 3

OUTPUT: MAGIC NUMBER = 42
          23   2  17
           8  14  20
          11  26   5


 INPUT: Enter order, first number, increment: 7, 90, 2

OUTPUT: MAGIC NUMBER = 966
         148 166 184  90 108 126 144
         164 182 102 106 124 142 146
         180 100 104 122 140 158 162
          98 116 120 138 156 160 178
         114 118 136 154 172 176  96
         130 134 152 170 174  94 112
         132 150 168 186  92 110 128
```

**3.7** A magic square is a matrix of distinct numbers with the same number of rows as columns and the sum of the numbers in each row, column, and diagonal is equal to the same total (the magic number). All magic squares with an odd "order" (number of rows and columns) can be generated using La Loubre's staircase method. General methods are still being explored for generating magic squares with an even "order" (number of rows and columns). Magic squares whose order is a multiple of four can be constructed by a particular method. However, the most difficult magic square to produce is one with an order of 6. The easiest method used to construct a 6 by 6 magic square is as follows:

1) Divide the 6 by 6 square into four 3 by 3 squares.

2) Using the "staircase" method to generate 3 by 3 magic squares, place the first nine numbers in the upper left 3 by 3 square, place the next nine numbers in the lower right-hand 3 by 3 square, place the next nine numbers in the upper right-hand 3 by 3 square, place the last nine numbers in the lower left-hand 3 by 3 square.

3) Transpose the three cells in positions (1,1), (2,2), (3,1) with those cells in positions (4,1), (5,2), and (6,1), respectively, where each position is designated by (Row, Column).

Write a program to display the 6 by 6 magic square using the staircase algorithm, given as input the first positive integer to use and the positive increment between each successive number. Each number is to be right justified within a four-character column. Begin the output by displaying the magic number (the sum of all the cells for a column, for a row, and for a diagonal).

For the example below, a 6 by 6 magic square starting with the number 1 and each successive number incremented by 1 would look like this after steps 1 and 2:

```
   8   1   6      26  19  24
   3   5   7      21  23  25
   4   9   2      22  27  20

  35  28  33      17  10  15
  30  32  34      12  14  16
  31  36  29      13  18  11
```

The final magic square is displayed below. Example:

```
     INPUT: Enter first number, increment: 1, 1

    OUTPUT: MAGIC NUMBER = 111
              35   1   6  26  19  24
               3  32   7  21  23  25
              31   9   2  22  27  20
               8  28  33  17  10  15
              30   5  34  12  14  16
               4  36  29  13  18  11
```

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

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

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

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

    OUTPUT:        *******         OUTPUT:        *******
                 **   *   **                    **NNN*AAA**
                *     *     *                  *NNNNN*AAAAA*
               *      *      *                *NNNNNN*AAAAAA*
              *       *       *              *NNNNNNN*AAAAAAA*
             *        *        *            *NNNNNNNN*AAAAAAAA*
             *        *        *            *NNNNNNNN*AAAAAAAA*
            *         *         *          *NNNNNNNNN*AAAAAAAAA*
            *         *         *          *NNNNNNNNN*AAAAAAAAA*
            *         *         *          *NNNNNNNNN*AAAAAAAAA*
            ***********         *          ***********AAAAAAAAA*
            *       **          *          *DDDDDDD**AAAAAAAAAA*
            *      *            *          *DDDDDD*AAAAAAAAAAAA*
            *    **             *          *DDDD**AAAAAAAAAAAAA*
             *  *              *            *DD*AAAAAAAAAAAAAA*
             *  *              *            *DD*AAAAAAAAAAAAAA*
              **              *              **AAAAAAAAAAAAAA*
               **            *                **AAAAAAAAAAAA*
                *           *                  *AAAAAAAAAAA*
                 **       **                    **AAAAAAA**
                   *******                        *******

    INPUT: (press any key)
```

**3.9** Write a program to produce THE UNIQUE ORDER of execution for jobs (that run Billing system programs) given a set of dependencies between the jobs. The program will first prompt for the number of dependencies (less than 8) that will be entered. Each input line of dependencies will consist of a 2-character job name that must finish before the second 2-character job, separated by a space. The output line must contain THE UNIQUE ORDERING of the jobs, all on one line, that will satisfy the dependencies. Examples:

```
     INPUT: Enter number of dependencies: 5
             Enter dependency: OA OU
            Enter dependency: OA OJ
            Enter dependency: OA OE
            Enter dependency: OJ OE
            Enter dependency: OE OU
    OUTPUT: JOBS MUST BE RUN IN THIS ORDER: OA OJ OE OU

     INPUT: Enter number of dependencies: 6
             Enter dependency: BK 5M
            Enter dependency: BE BK
            Enter dependency: BM BN
            Enter dependency: 5M BN
            Enter dependency: BK BM
            Enter dependency: BM 5M
    OUTPUT: JOBS MUST BE RUN IN THIS ORDER: BE BK BM 5M BN
```

**3.10** The digits 123456789 can be rearranged to form a nine-digit perfect square with unique digits. For example, swapping 7 with 8, 6 with 9, 4 with 8, 3 with 7, 2 with 4, and 1 with 8, forms the perfect square 847159236 (the square of 29106). This square is formed by making six exchanges:

```
     swap 7 with 8          123456879
     swap 6 with 9           123459876
     swap 4 with 8          123859476
     swap 3 with 7          127859436
     swap 2 with 4          147859236
     swap 1 with 8          847159236
```

Write a program to find the nine-digit perfect square that requires the fewest exchanges of pairs of digits from the original 123456789 number. Display the square with its square root followed by the number of exchanges required to form the square. The format of the output must be two lines as follows in the first example with # representing a digit. The second example output shows what the output would be like IF the fewest number of exchanges is actually 6, but it is fewer. Example outputs:

```
     OUTPUT: ######### IS THE SQUARE OF #####
             AND WAS FORMED BY EXCHANGING # PAIRS OF DIGITS

     OUTPUT: 847159236 IS THE SQUARE OF 29106
             AND WAS FORMED BY EXCHANGING 6 PAIRS OF DIGITS
```