Great Computer Challenge 2008

Scientific/Non-Business Programming

Level IV

Problem 1

Greek Numbers

The Greeks began an examination of numerology by classifying all positive integers as either perfect, abundant, or deficient. This classification scheme is based on the factors (even divisors) of the number. If the sum if all of the factors of a number (excluding the number itself) equals the number then it is said to be "perfect." For example, the factors of 6 are 1, 2, 3, and 6. Therefore, the number 6 is a perfect number since the sum of the factors of 6 (excluding 6) is 1 + 2 + 3 = 6. An abundant number is one in which this sum of factors (excluding the number itself) is greater that the number. An example of an abundant number is 12 because 1 + 2 + 3 + 4 + 6 = 16 > 12. All numbers that are neither perfect nor abundant are deficient.

This program accepts a sequence of positive integer values and classifies each as abundant, perfect, or deficient. The program also displays the factors of the number. Input is terminated by an input value of zero.

INPUT
Prompted input consists of nonnegative integer values. An input of zero (0) signals termination.

OUTPUT
Each input value is prompted by the following output line:

Specify a positive integer:

Following the input a line like the following is output, where < type > is either "abundant," "perfect," or "deficient" as is appropriate for the input value:

This number is < type > (factors given below).

ERRORS
All input is assumed to be correct. Any incorrect input produces undefined results.

EXAMPLE
Below is a sample execution (all input in italics).

Specify a positive integer: *6*
This number is perfect (factors given below).
1 2 3

Specify a positive integer: *12*
This number is abundant (factors given below).
1 2 3 4 6

Specify a positive integer: *333*
This number is deficient (factors given below).
1 3 9 37 111

Specify a positive integer: *0*

Problem 2

Goody, the Clever Ant

Assume an ant is crawling over a rectangular grid consisting of $M \times N$ cells. Each cell in the grid is marked with one of {.,L,R,S}. Goody can move from a cell to any one of its four neighbours subject to the following conditions: From a cell marked with a ".", the ant can move in any direction. From a cell marked with an "L" he must move left with respect to the direction he was moving when he entered this cell. From a cell marked with an "R", he must move right with respect to the direction he was moving when he entered this cell. If a cell is marked "S" his crawl comes to a stop. You may assume that all the cells on the boundary are marked with ".", so that it is always possible to move following the rules.
Given a description of the grid and the cell where the ant begins his (you may assume that this cell is always marked with a ".") your task is to determine the minimum number of steps that the ant needs to take before he reaches a cell marked with "S".
For example, here is a 7 ×8 grid. The ant starts at the cell (4,1) - that is, the cell in row 4, column 1. He can reach the "S" labeled cell at (2,4) by following the sequence (4,1), (3,1), (2,1), (1,1), (1,2), (1,3), (1,4), (2,4) and this takes 7 steps. He can also reach that cell in 5 steps by using the following shorter sequence: (4,1), (4,2), (4,3), (3,3), (2,3), (2,4). You can check that this is indeed the shortest such sequence to any cell labeled with a S.

```
. . . . . . . .

. L R S . L . .

. . . . . . L .

G . L . . R . .

. . R . R . . .

. . . . . . S .

. . . . . . . .
```

Input Format
The first line of the input consists of two integers M and N indicating the number of rows and columns in the grid. The next M lines contain N characters, each of which is . or L or R or S or G. There is exactly one cell in the grid marked with a A and this is the cell from with the ant begins crawling.

Output Format
If the ant can crawl to a cell labeled S, print a single integer indicating the minimum number of steps required to reach *some* cell labeled with S. Otherwise print -1.

Test Data
You may assume that $N \leq 60$ and $M \leq 60$ in 80% of the inputs. In all the inputs $N \leq 1000$ and $M \leq 1000$.

Example
We now illustrate input and output formats using the example described above.
Sample input

```
7 8
........
.LRS.L..
......L.
G.L..R..
..R.R...
......S.
........
```

Sample output

```
5
```

Problem 3

A Spelling Problem

A new systems analyst is working on a project of text assistance in the area of validating plurals of words for a spelling checker. He needs a program that will accept a word and produce the correct plural of the word.

Write a program that will accept an English word and produce the plural of the word. It is noted that some special conditions exist for valid English pluralization:

1. if the last letter of the word is an 'h' and the next to last letter is either a 'c' or an 's', then the plural of the word adds an 'es' to the word.

2. if the last letter of the word is an 's', then the plural of the word adds an 'es' tot he word.

3. if the last letter of the word is a 'y', then if the next to last letter of the word is a vowel, then the plural adds an 's' to the word.

4. if the last letter of the word is a 'y', and it is not preceded by a vowel, then the plural replaces the 'y' with an 'ies'

5. if the last letter of the word is a 'z' that is preceded by a vowel, then the plural adds a 'zes' to the word.

6. if the last letter of the word is a 'z' that is not preceded by a vowel, then the plural adds an 'es' to the word.

7. most all other conditions add an 's' to the word to form the plural of the word.

Problem 4

Sounds the Same

The soundex algorithm is used by spell checking programs to suggest words that sound like the word that the user typed. It works by encoding the original word. If a word in the spell check dictionary has the same code then it is a word to suggest to the user.

Your program will take as input two words. It will calculate the soundex encoding for each. Then it will tell whether the two words have the same soundex encoding. Finally it will tell the length of the common prefix.

The Algorithm

1. The first character in the encoding is the first letter of the word.

2. The following encoding is used on subsequent letters

   a. discard all vowels and vowel-like letters: a,e,i,o,u,h,w,y

   b. The remaining letters are mapped to digit characters:

      b,p => 1
      f,v => 2
      c,k,s => 3
      g,j => 4
      q,x,z => 5
      d,t => 6
      l => 7
      m,n => 8
      r => 9

   c. if two or more of the same digit are next to each other, discard all but the first.

Examples

input: *APPLESAUCE APPLAUSE*

output:
soundex of APPLESAUCE is A173
soundex of APPLAUSE   is A173
they have the same encoding
they share a prefix of length 4

----

input: *BANANABOAT BANNER*

output:
soundex of BANANABOAT is B816
soundex of BANNER is B89
they do not have the same encoding

they share a prefix of length 2

----

input: *MOTHER FATHER*

output:
soundex of MOTHER is M69
soundex of FATHER is F69
they do not have the same encoding
they share a prefix of length 0

Problem 5

A Basic Calculator

You are to accept as input an integer B where 0 < B <10, this value is to be the base for a calculator that performs addition and subtraction on values in that system. If a users enter an illegal values they should be prompted for new ones.

Example
Input > *4*
(this means a base 4 calculator so the allowed values corresponding to base 10 values of 0 through 10 would be 0, 1, 2, 3, 10, 11, 12, 13, 20, 21, 22, 23)

Menu
(A)dd two base 4 values
(S)ubtract two base 4 values
(Q)uit

Input > *A*
Output > "first value?"
Input > *51*
Output > "sorry, not base 4 try again"
Input > *13*
Output > "second value?"
Input > *23*
Output > "the sum of 13 and 23 is 102 in base 4"