Fourth Southern African Regional ACM Collegiate Programming Competition

Sponsored by IBM

Problem Set

Fourth Southern African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

Problem 1 – Pink balloon
Radix 3

The Great Sand Council (GSC) of the planet Phleebutt (apologies to Sierra) have devised a base 3 number system to suit their physiology. The symbols used to represent the three valid digits of their number system are '0', '1', and '-', following the unusual configuration of their 'hands' (don't ask). The decimal counterparts of the three symbols are, respectively, 0, 1 and -1.

Each position in a number has a value three times greater than the position immediately to its right. For example, the number '10-' has the value 8 in decimal, since 1*9 + 0*3 + -1*1 = 8. Similarly, the number '-1' has the decimal value -2, since -1*3 + 1 = -2.

You have to write a program that can convert 32-bit signed decimal integers into their equivalent Great Sand Council (GSC) representations.

Sample Input

Your program will receive a list of integers as input, for example:

```
10
2
-17
42
1024
```

Sample Output

You have to echo the input numbers, followed by their GSC representations (include all extra symbols, as shown below):

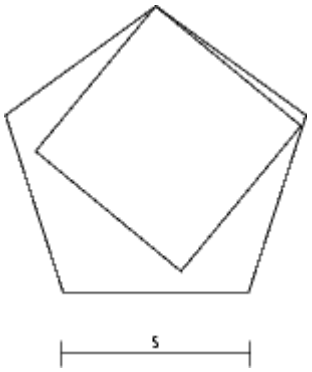
```
10 = 101 GSC
2 = 1- GSC
-17 = -1-0- GSC
42 = 1---0 GSC
1024 = 111-0-1 GSC
```

Fourth Southern African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

Problem 2 – Yellow balloon
Fitting Squares in Pentagons

You have obtained a set of pentagonal titanium plates, formerly used in the heat-exchanger of a nuclear power station. From this you want to construct cubic boxes (small safes?), for which you will need square plates. The problem is that you don't have the equipment to forge the pentagonal plates into neat squares. The best you can do is to try and cut the largest possible square from each of the pentagonal plates.



Thus, given the side length of the pentagon, s, you must compute the side length of the largest square that will fit inside it. You may assume that the one corner of the square always coincides with one of the corners of the pentagon, as shown in the diagram. (Clearly the square shown in the diagram is not the largest possible one.)

Sample Input

Your input will be a list of pentagon side-lengths, one per line.

```
0.237788
0.291066
0.845814
0.152208
0.585537
```

Sample Output

Your output must be the maximal square side-length, one per line. Your answers must be rounded to 10 decimal places (after the decimal period). The following results were obtained from the input listed above.

```
0.2538138844
0.3106825915
0.9028182111
0.1624661619
0.6249996653
```

Fourth Southern African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

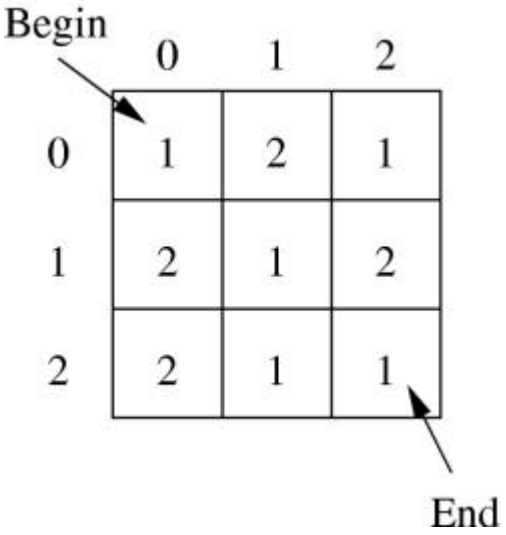
Problem 3 – Green balloon
Trampolines

Our intrepid adventurer, Miss Croft, has stumbled across a difficult puzzle while exploring an ancient Mayan temple. The puzzle comprises a grid of trampolines contained in a rectangular room. Each trampoline can be represented by a square on a grid, as shown below. The difficulty in crossing the room lies in the fact that each trampoline has a different degree of elasticity, so that some trampolines will propel you only one square, while others may propel you several squares.

To safely reach the exit, Lara has to jump on the trampolines in the correct sequence in order to reach the `end' square. To simplify matters, the `Begin' square will always be at position (0,0) on the grid, while the `End' square will be at the diagonally opposite corner of the grid, e.g. at (2,2) in the diagram below. The coordinates are listed in pairs of the form (horizontal, vertical).

Several rules apply:

1. You may only jump along the major axes of the grid, i.e. in the diagram you may only jump horizontally or vertically, not diagonally.
2. You must jump the exact number of squares indicated by the trampoline's strength. Using the grid shown below, you can (for example) jump from (1,0) to (1,2).
3. You shouldn't jump into the walls at the edge of the grid. For example, you cannot try to jump from (1,0) in the direction of (2,0), since you would hit the wall. The (1,0) square forces you to jump a distance of 2 squares, the wall is only one square away in that direction.
4. You may visit each square once at most.



Your task is to list the shortest possible route from the starting trampoline (square) to the end trampoline (square). The shortest possible route is defined as the route requiring the fewest jumps.

Sample Input

Each block of input consists of a line containing the dimensions of the grid, followed by the grid itself.

```
3 3
1 2 1
2 1 2
2 1 1
```

The input may contain several such blocks (each one a different puzzle). The grid dimensions are given in the same order as all the other coordinates, i.e. horizontal dimension first.

All problems you receive will have a valid solution.

Sample Output

Your program must output the shortest route from the `Begin' square to the `End' square for each puzzle. The solutions must be listed one per line as a sequence of coordinates in the order that they were visited. The output for the sample input above would be:

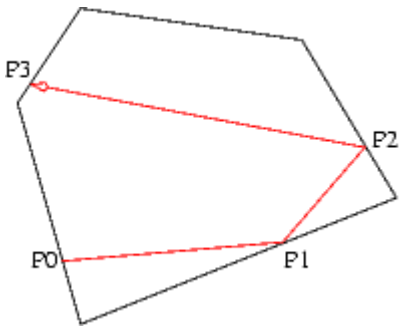
```
(0,0) (1,0) (1,2) (2,2)
```

Fourth Southern African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

Problem 4 – Purple balloon
Mirrors

You have been asked to design a security system for protecting the crown jewels. After careful consideration (and watching numerous Remington Steele episodes), you have decided to set up a laser tripwire alarm system. The system works by bouncing a laser beam off mirror strips fitted to the walls of the room. Since the beam will follow a fixed path through the room, you can place a single sensor at the end of the laser's path. If anyone breaks the beam at any point along the path that the laser follows, the light will not reach the sensor, setting off the alarm. The diagram below illustrates a how a beam originating at P0 is reflected off the walls, eventually reaching the point P3 (this is a top-down view of the room).



Your task is to calculate the path that the beam will take through the room so that you know where you have to place the sensor. Your input will be the geometry of the room, and the origin of the laser beam. Note that the problem domain is restricted to two dimensions.

The laser beam is reflected in a direction so that the angle of reflection (with respect to the wall) is equal to the angle of incidence. You don't have to worry about hitting corners; test cases will never have beams hitting the corner exactly.

Input

Your input consists of the following elements:

```
number_of_reflections
laser_x laser_y
laser_direction_x laser_direction_y
number_of_walls
wall_start_x wall_start_y wall_stop_x wall_stop_y
```

All the values are real numbers, except the number_of_reflections and number_of_walls parameters, which are integers.

You have to follow the path of the beam originating at position `(laser_x, laser_y)` travelling in direction `(laser_direction_x, laser_direction_y)`. Note that the laser direction is a vector, not a point.

Each problem will have `number_of_walls` wall specifications; only the first one is shown above. Your input may contain several problem specifications, each following the format defined above.

Output

Your output will be the coordinates (accurate to 2 decimal places) of the point where the beam strikes on its nth reflection, where n = `number_of_reflections`. For example, in the diagram above, your output will be the coordinates of point P3, with `number_of_reflections` = 3 (thus the endpoint is treated as part of the reflection count). Repeat this for every problem specified in the input.

Sample Input

```
5
0 0
1 0.5
4
-1 -1  1 -1
 1 -1  1  1
-1  1  1  1
-1  1 -1 -1
```

Sample Output

```
0.00 -1.00
```

Fourth Southern African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

Problem 5 – Blue balloon
Word Sums

You've all seen this type of puzzle before: two numbers and their sum are expressed as letters of the alphabet, and you have to find their numeric values. For example

```
  hello
+ there
-------
  world
```

can be solved to yield

```
  56442
+ 15606
-------
  72048
```

In other words, to each letter of the alphabet found in the original puzzle you must assign a digit 0 through 9 so that the resulting sum is correct. The following rules apply:

1. Each number (0-9) may only be assigned to a single letter, i.e. if 5 is assigned to 'a', then no other letter may have the value 5.
2. The leading digit of any of the three terms may not be zero.
3. All values you are presented with will fit in a 32-bit variable, i.e. the longest word will have less than 10 letters.

Some of the puzzles may have more than one solution. In these cases, you have to give preference to the solution with the smallest digit in the least significant position of the first addend. If there's more than one such solution, you must prefer the one with the smallest digit in the least significant position of the second addend. Should there still be more than one solution, you pick the one with the smallest digit in the second least significant (10s) position of the first addend, and so on.

Return only one solution, that is, the one complying with the above constraints.

Some sums will have no valid solutions. In these cases, simply print out the phrase "No solution".

Sample Input

Each set of three words defines a puzzle which you must solve. Typical input might look like:

```
hello there world
i am nuts
```

Sample Output

You have to echo the input words, followed by the solution, using the format shown below (including the arrow and the other symbols):

```
hello + there = world -> 56442 + 15606 = 72048
i + am = nuts -> No solution
```

Note that the first example (hello there world) has many solutions. The one shown above conforms to the constraints specified above.

Fourth Southern African Regional
ACM Collegiate Programming Competition

Sponsored by IBM

Problem 6 – Red balloon
Counting Words

As part of a new encryption algorithm that you are designing, you are required to find a way of uniquely numbering all possible words. Your alphabet is restricted to the lowercase letters "a" through "z". The system that you have designed ranks the words first by length, and then alphabetically. For example:

```
a 1
b 2
z 26
aa 27
ab 28
zz 702
...
physics 5,049,467,949
```

Your task is to write a program that, given a word, prints out the numeric representation of this word. All words will be shorter than 41 characters.

Sample Input

Your input will consist of a sequence of words, one per line.

```
snowfall
elementary
transcendental
superstructural
```

Sample Output

You have to echo each input word, followed by its numeric representation. The numbers must be printed in the format shown below, with a comma separating each set of three digits.

```
snowfall 157,118,051,752
elementary 29,697,684,282,993
transcendental 51,346,529,199,396,181,750
superstructural 1,279,341,593,224,884,122,582
```