Unimelb Code Masters 2015

Rules and Regulations

- Teams consist of two or three students, and may use up to two computers.
- Problems and data sets will be distributed on USB stick at the start of the competition, thus your computing devices must have USB capability. For online participants, teachers/supervisors will be supplied with problems and data sets via web browser to put onto their own local USB just prior to the start of the competition.
- Communication devices may not be used in any way during this competition. This includes Internet-enabled devices (switch off wifi on your laptops). You may not use a mobile phone for any purpose, including as a dictionary, calculator, etc.
- Judges and supervisors may inspect any data on any computing devices that are used by teams before, during and after the competition. If you have sensitive data that you do not want the judges to see, do not bring it with you to the competition.
- Teams that attempt to circumvent the "no-internet" rule by pre-downloading online material and storing it locally (for example, libraries of past problems and solutions in similar competitions, wikipedia pages, and so on) will be penalised at the judges' discretion.
- Reference material directly related to your current school studies is permitted. Teachers accompanying teams may be asked to verify that data on a student's computing device is related to their school work, and has not been downloaded specifically for the competition.
- At any time during the competition, one member of a team may approach the judges' table (supervisor for online participants) to submit an answer to one or more questions of a problem. The judges will indicate if the answer is correct, or not. Only three submissions per question in a problem are allowed, so if you get the first two submissions incorrect, go to the next question in the problem until you are confident in your code.
- Each team will be ranked at the end based on their answers, with ties being broken based on the methods of obtaining the solution. For tie breaking purposes, hand calculation will receive the lowest marks, and efficient algorithms will receive the highest. As such, teams and their code must be available for inspection by the judges during the judging process.
- Generally, the first one or two questions in a problem can be done by hand; but later questions within a problem will require some computation.
- The judges' decision is final. No discussion will be entered into.
- The top three teams in the junior (7-9) and senior (10-12) divisions will be awarded prizes.

DO NOT TURN THE PAGE UNTIL INSTRUCTED TO DO SO

1 Square Roots

The Newton-Raphson method allows for successive approximations to a function's value. In particular, if the first guess at the $\sqrt{k}$ is x, then a better guess is $\frac{1}{2}(x + k/x)$. This equation can be applied over and over, improving the approximation each time. More formally, the i'th guess at $\sqrt{k}$ can be written as

$$x_i = \begin{cases} k, & \text{if } i = 1 \\ \frac{1}{2}(x_{i-1} + k/x_{i-1}), & \text{otherwise} \end{cases}$$

Using this equation to compute $\sqrt{100}$, for example, gives

$$\begin{aligned} x_1 &= 100.00 \\ x_2 &= 50.50 \\ x_3 &= 26.24 \\ x_4 &= 15.03 \\ x_5 &= 10.84 \\ x_6 &= 10.03 \\ x_7 &= 10.00 \end{aligned}$$

In the following questions, give your answer to 2 decimal places.

1.1 Question 1

What is x_2 if $k = 1000$?

1.2 Question 2

What is x_3 if $k = 1000$?

1.3 Question 3

What is x_9 if $k = 1000$?

1.4 Question 4

What is x_{10} if $k = 1234.56789$?

1.5 Question 5

What is x_{100} if $k = 1234.56789$?

2 Water pools

Imagine a 2D slice of a landscape, shown on its side like so:

```
        #
 #      #
 ##  # ##
### ####
#########
```

The # characters indicate rocky ground. The vertical dimension shows the height of the rocks.

Suppose that it rains heavily and the rocky landscape fills up with water like so (the '-' character indicates water):

```
        #
 #------#
 ##--#-##
###--####
#########
```

The height of the rocks in the landscape dictates how high the water can fill before it starts spilling over the sides.

In the above example we can see that the maximum amount of water that can be contained in the landscape is 11 units (there are 11 '-' characters in the diagram, each representing a unit of water).

Note that even though the rightmost rock is 5 units high, the example landscape cannot retain any more than 11 units of water because the leftmost highest rock is only 4 units high. Any more water would spill over the edge.

In this problem we will represent the landscape as a sequence of non-negative integers, such that each integer represents the height of a column of rock from left-to-right. The example landscape above would be represented like so:

```
2
4
3
1
1
3
2
3
5
```

Your task is to write a program which will read in a description of a rocky landscape as described above and print out the maximum number of units of water that can be retained in the landscape. The answer will be a non-negative integer.

You may assume that the input will contain no more than 1000 integers. The answer will fit into an unsigned 32 bit integer.

Below is another example:

```
       #
  #----#
 ##----#
###----#
####---##----#
####--####---##
####-#######-###
```

The above landscape can hold a maximum of 26 units of water. The input would be described like so:

```
4
5
6
3
0
1
2
7
3
2
1
0
1
3
2
```

2.1 Question 1

How much water can the landscape in the file `water_pools_1.txt` hold?

2.2 Question 2

How much water can the landscape in the file `water_pools_2.txt` hold?

2.3 Question 3

How much water can the landscape in the file `water_pools_3.txt` hold?

2.4 Question 4

How much water can the landscape in the file `water_pools_4.txt` hold?

2.5 Question 5

How much water can the landscape in the file `water_pools_5.txt` hold?

3 Langton's Ant

Imagine a simple simulation of an ant on a rectangular two-dimensional grid of black and white square cells.

The ant walks about the grid one step at a time, obeying the following two rules:

1. If the ant is standing on a white cell, it changes the cell to black, turns right ninety degrees and takes one step forward.
2. If the ant is standing on a black cell, it changes the cell to white, turns left ninety degrees and takes one step forward.

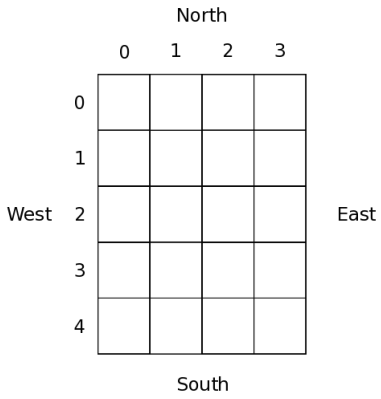
If the ant reaches any of the edges of the grid it moves to the cell on the opposite side. In other words its movement wraps around at the edges of the grid.

Your task is to write a program to simulate the movement of the ant, given the following inputs:

- The height and width of the grid (number of rows and columns of cells): $1 \leq$ height ≤ 1000, $1 \leq$ width ≤ 1000.
- The starting coordinate of the ant: $0 \leq$ column $\leq$ height-1, $0 \leq$ row $\leq$ width-1.
- The number of steps to simulate: $1 \leq$ steps ≤ 100000.

Grid coordinates are zero-based. So the top-left coordinate is at row 0 and column 0. If R is the number of rows and C is the number of columns in the grid then the bottom right coordinate is at row R-1 and column C-1. The grid is oriented according the the cardinal directions north, east, south and west. Row 0 is the northernmost row and row R-1 is the southernmost row. Column 0 is the westernmost column and column C-1 is the easternmost column.

Below is a diagram of grid with 5 rows and 4 columns.



All the cells in the initial grid are white.

The input to your program will be 5 lines of text:

1. the number of rows in the grid.
2. the number of columns in the grid.
3. the starting row of the ant.
4. the starting column of the ant.
5. the orientation of the ant, one of `N`, `E`, `S` or `W`, representing the four cardinal directions north, east, south and west.
6. the number of steps of ant movement to simulate.

You may assume that all the lines in the input are valid according to the problem specification.

Your program should simulate the movement of the ant over the number of specified steps and print the sum of the final row and column of the ant.

For example if the input is like so:

```
5
4
3
2
W
5
```

your program should print `6` as its answer, as the ant ended up at row 4 column 2.

3.1 Question 1

What is the result of your program on `ant1.txt`?

3.2 Question 2

What is the result of your program on `ant2.txt`?

3.3 Question 3

What is the result of your program on `ant3.txt`?

3.4 Question 4

What is the result of your program on `ant4.txt`?

3.5 Question 5

What is the result of your program on `ant5.txt`?

4 Melbourne Tram Network

The following five questions all relate to the data in the single `trams.txt` file which contains lines describing tram stops on tram routes in Melbourne. Each line has two numbers separated by a space. The first is a stop number and the second is a route number. For example, the first 5 lines are

```
2382 11
1440 19
2330 57
3557 30
2106 5
```

No stop number has more than 4 digits, and no route number has more than 3 digits. A stop may be shared by multiple routes.

4.1 Question 1

How many different routes are there in the network?

4.2 Question 2

How many different tram stops are there in the network?

4.3 Question 3

Which tram route has the most stops?

4.4 Question 4

What is the maximum number of routes that go through any one tram stop?

4.5 Question 5

What is the maximum number of stops that are shared by any two tram routes?

5 Perfect Shuffle

Ace McPoker prides herself on the ability to shuffle two decks of cards perfectly, so that the cards from the two decks strictly alternate when she is finished. For example, if she has two very small decks as follows

```
Deck 1: AS KS QS TS 9S
Deck 2: 2H 3H 4H 5H 6H
```

then the resulting shuffle will be

```
    AS 2H KS 3H QS 4H TS 5H 9S 6H
```

In this schematic, each card is represented by two characters, the first being one of `A23456789TJQK` representing the value of the card, and the second one of `CDHS` representing the suit of the card.

When Ace shuffles, the first card in the resulting deck is always from Deck 1, and the second from Deck 2. The decks always contain the same number of cards, so the last card is always from Deck 2.

In each of the questions below, assume that Ace begins with two full decks of cards that are in the order of all the clubs first (`C`), then the Diamonds (`D`), then the Hearts (`H`), then finally the Spades (`S`). Within each suit, the cards are ordered `A23456789TJQK`. Hence in each deck the `AC` is in position 1, the `5C` is in position 5, and the `KS` is in position 52.

Positions are counted from 1 being the first card in the deck, and can range from 1 to 104 in the final shuffled deck, and 1 to 52 in the starting deck.

5.1 Question 1

After her first shuffle, what is the lowest position of a Jack of Hearts (`JH`)?

5.2 Question 2

After her first shuffle, what is the lowest position of a 4 of Diamonds (`4D`)?

5.3 Question 3

After her first shuffle, Ace splits the shuffled deck exactly in half so that the first 52 cards form Deck 1, and the second 52 cards form Deck 2. If she then shuffles these decks once, what is the lowest position of a King of Diamonds in the resulting deck?

5.4 Question 4

If, beginning with the two starting decks, Ace shuffles 5 times, exactly splitting the decks into two new decks of 52 as in Question 3, what is the lowest position of a Queen of Clubs after the 5 shuffles?

5.5 Question 5

What is the minumum number of shuffles after the first that is required to get a King of Spades back into position 52?

6 Jumping up stairs

Ferdinand is a fitness instructor. One of his favourite exercises is to jump up flights of stairs.

He wonders how many different ways he can jump up a flight of N stairs if he chooses between 1, 2 and 3 stairs at each jump. He must start at the bottom of the stair case and always land exactly on the top stair with the final jump.

He works out the easy cases in his head. For example, if there is only 1 stair there is only 1 way to do it. If there are 4 stairs then there are 7 solutions, as shown below where each line shows a possible sequence of jumps that sums to 4.

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

Write a program to help Ferdinand calculate the solution for larger values of N. The input to your program is an integer number of steps between 1 and 40. The output should be the number of possible ways Ferdinand can jump up the stairs.

6.1 Question 1

How many possible ways can Ferdinand jump up 5 stairs?

6.2 Question 2

How many possible ways can Ferdinand jump up 6 stairs?

6.3 Question 3

How many possible ways can Ferdinand jump up 10 stairs?

6.4 Question 4

How many possible ways can Ferdinand jump up 29 stairs?

6.5 Question 5

How many possible ways can Ferdinand jump up 35 stairs?