University of Central Florida

2017 (Fall) “Practice” Local Programming Contest

Problems

Problem#	Difficulty Level	Filename	Problem Name
1	Easy	coin	Good Coin Denomination
2	Easy	smart	Sorry About That, Chief!
3	Medium	matrix	Matrix Transformation
4	Medium	polycake	Cut the Cake!
5	Hard	sorted	Doubly Sorted Grid
6	Hard	marbles	Marbles

Call your program file: filename.c, filename.cpp, filename.java, or filename.py

For example, if you are solving Good Coin Denomination:

Call your program file: coin.c, coin.cpp, coin.java, or coin.py

Good Coin Denomination

filename: `coin`
(*Difficulty Level:* `Easy`)

Different countries use different coin denominations. For example, the USA uses 1, 5, 10, and 25. A desirable property of coin denominations is to have each coin at least twice the amount of its previous coin in sorted order. For example, the USA denominations have this property, but the coin denominations {1, 5, 6} do not (6 is not at least twice 5).

The Problem:

Given the coin denominations, you are to determine if the set has the above property.

The Input:

The first input line contains a positive integer, *n*, indicating the number of denomination sets to check. The sets are on the following *n* input lines, one set per line. Each set starts with an integer d $(1 \leq d \leq 10)$, which is a count of various coin amounts in the set; this is followed by *d* distinct positive integers (each less than 1,000) giving each coin amount (assume the coin amounts are given in increasing order).

The Output:

For each set of coin denominations, output a single integer on a line by itself indicating whether or not the set has the above property. Output the integer 0 (zero) if the set has the property and 1 (one) if it does not.

Sample Input:

```
2
4 1 5 10 25
3 1 5 6
```

Sample Output:

```
0
1
```

UCF “Practice” Local Contest — Aug 26, 2017

Sorry About That, Chief!

filename: `smart`
(*Difficulty Level:* `Easy`)

When Dr. Orooji was your age, one of the popular TV shows was “Get Smart!” The main character in this show (Maxwell Smart, a secret agent) had a few phrases; we used one such phrase for the title of this problem and we’ll use couple more in the output description!

The Problem:

A “prime” number is an integer greater than 1 with only two divisors: 1 and itself; examples include 5, 11 and 23. Given a positive integer, you are to print a message indicating whether the number is a prime or how close it is to a prime.

The Input:

The first input line contains a positive integer, *n* $(n \leq 100)$, indicating the number of values to check. The values are on the following *n* input lines, one per line. Each value will be an integer between 2 and 10,000 (inclusive).

The Output:

For each test case, output two integers on a line by itself separated by a space. The first integer is the input value and the second integer is as follows:

- If the input number is a prime, print 0 (zero). This refers to Maxwell Smart phrase: “`Would you believe it; it is a prime!`”
- If the input number is not a prime, print the integer `d` where `d` shows how close the number is to a prime number (note that the closest prime number may be smaller or larger than the given number). This refers to Maxwell Smart phrase: “`Missed it by that much (d)!`”

Sample Input:

```
4
23
25
22
10000
```

Sample Output:

```
23 0
25 2
22 1
10000 7
```

UCF "Practice" Local Contest — Aug 26, 2017

Matrix Transformation

filename: `matrix`

(*Difficulty Level:* `Medium`)

You have an integer matrix A, with R rows and C columns. That means it has R rows with each row containing C integers. Two integers are adjacent if their container cells share an edge. For example, in the following grid

0	1	2
3	4	5
6	7	8

(0, 1), (4, 5), (1, 4), (5, 2) are adjacent but (0, 4), (2, 6), (5, 7) are not adjacent.

You are allowed to do only one kind of operation in the matrix. In each step you will select two adjacent cells and increase or decrease those two adjacent values by 1, i.e., both values are increased by 1 or both values are decreased by 1.

The Problem:

Given a matrix, determine whether it is possible to transform it to a zero matrix by applying the allowed operations. A zero matrix is the one where each of its entries is zero.

The Input:

The first input line contains a positive integer, *n*, indicating the number of matrices. Each matrix starts with a line containing R ($2 \le R \le 30$) and C ($2 \le C \le 30$) separated by a single space. Each of the next R lines contains C integers. Each of these integers is between -20 and +20 inclusive. Assume that each input matrix will have at least one non-zero value.

The Output:

For each matrix (test case), output a single integer on a line by itself indicating whether or not it can be transformed to a zero matrix. Output the integer 0 (zero) if the matrix can be transformed to a zero matrix and 1 (one) if it cannot.

Sample Input:

```
6
3 3
-2 2 2
1 1 0
2 -2 -2
3 3
-1 0 1
-2 -1 1
0 1 2
3 3
-1 0 1
0 2 -1
-1 1 2
3 3
-1 2 1
-1 -1 -3
1 1 -1
2 3
0 -2 3
1 3 1
2 3
3 1 1
2 0 1
```

Sample Output:

```
0
1
1
0
1
0
```

UCF “Practice” Local Contest — Aug 26, 2017

Cut the Cake!

filename: `polycake`

(*Difficulty Level:* `Medium`)

On the faraway planet of Gastronomia, the native Gastronomes consider polygonal pancakes to be the highest form of art (as well as the tastiest). Every year, they celebrate the Polycake Festival, in which thousands of ninja chef monasteries perform the Thousand Slices, a series of rituals of tremendous importance to Gastronome society. In each of these rituals, a polycake is carefully placed on a ceremonial altar, and a ninja chef will solemnly cut the cake with a single, perfectly straight slice, dividing it into two pieces, one on the North part of the altar (representing the past) and the other on the South part (representing the future). The two pieces are carefully separated, and their perimeters measured. The results will determine the proportions of the ingredients used in polycakes planetwide, until the next Polycake Festival.

As the foremost Novice at the Temple of the Promised Cosmic Polycake, you are entrusted the task of measuring the perimeters of the two slices. Take care, for any mistakes will be mercilessly mocked by the Brotherhood of the Illusory Polycake[1]. It would be best if you wrote a program to do this.

The Problem:

Given a polygon representing the polycake in the Cartesian plane and a line parallel to the X-axis indicating the position of the slice, you are to compute the perimeter of each of the two pieces thus formed. The perimeter of a polygon is defined as the sum of the lengths of all its sides. Assume that the input polygons will be convex (i.e., not concave) and simple (i.e., not complex, not intersecting).

The Input:

The first input line contains a positive integer, *n*, indicating the number of polycakes used in the ritual. This is followed by *n* data sets, each representing a single slicing of a polycake.

The first line of each set consists of two integers, V $(3 \le V \le 10)$ and Y $(-1000 \le Y \le 1000)$, representing the number of vertices in the polycake and the y-coordinate of the horizontal cut, respectively. (Recall that the equation of a line parallel to the X-axis is of the form $y=k$.)

The next V lines each contain two integers, x and y $(-1000 \le x, y \le 1000)$, the Cartesian coordinates of the vertices of the polycake, in counter-clockwise order.

[1] Reviled heretics who claim that the Cosmic Polycake is a lie.

To minimize complications (and following the rules of the ritual), it is guaranteed that the cut will always go through the cake and will never pass through a vertex.

The Output:

For each test case, output one line. That line should contain "`a b`", the perimeters of the two slices, in increasing order. Output the results to 3 decimal places, rounded to the nearest thousandth (e.g., 77.0113 should round to 77.011, 88.0115 should round to 88.012, and 99.0117 should round to 99.012).

Sample Input:

```
2
4 2
0 0
4 0
4 4
0 4
6 10
3 15
10 1
12 5
11 19
9 23
6 20
```

Sample Output:

```
12.000 12.000
25.690 35.302
```

Doubly Sorted Grid

filename: `sorted`
(*Difficulty Level:* `Hard`)

A rectangular grid with lower case English letters in each cell is called *doubly sorted* if in each row the letters are non-decreasing from the left to the right, and in each column the letters are non-decreasing from the top to the bottom. In the following examples, the first two grids are doubly sorted, while the other two are not:

```
abc    ace    aceg    base
def    ade    cdef    base
ghi    bdg    xxyy    base
```

The Problem:

You are given a partially-filled grid, where some of the cells are filled with letters. Your task is to compute the number of ways you can fill the rest of the cells so that the resulting grid is doubly sorted. The answer might be a big number; you need to output the number of ways modulo 10,007.

The Input:

The first line of input gives the number of test cases t $(1 \le t \le 40)$. The t test cases follow. Each test case starts with a line containing two integers, r $(1 \le r \le 10)$ and c $(1 \le c \le 10)$, the number of rows and the number of columns respectively. This is followed by r lines, each containing a string of length c, giving the partially-filled grid. Each character in the grid is either a lower-case English letter, or '.' indicating that the cell is not filled yet.

The Output:

For each test case, output one line. That line should contain the number of possible doubly-sorted grids, modulo 10,007.

Sample Input:

```
3
2 2
ad
c.
3 3
.a.
a.z
.z.
4 4
....
.g..
.cj.
....
```

Sample Output:

```
23
7569
0
```

UCF “Practice” Local Contest — Aug 26, 2017

Marbles

filename: `marbles`
(*Difficulty Level:* `Hard`)

You have *2n* marbles on a square grid. The marbles are colored in *n* different colors such that there are exactly 2 marbles of each color. The marbles are placed at the coordinates (1,0), (2,0), ..., (*2n*, 0).

Your task is to draw a path for each color that joins the two marbles of that color. Each path should be composed of vertical or horizontal line segments between grid points. No two paths can intersect or touch each other. No path may cross the y=0 line. Each path can only touch the y=0 line at the position of the two marbles it is connecting, so the first and last line segment of each path must be vertical.

The Problem:

Given an arrangement of marbles, return the minimum height of a solution, or return -1 if no solution exists. The height is defined as the difference between the highest and lowest Y-coordinates of the paths used.

An example:

```
red red blue yellow blue yellow
```

One solution would be:

```
 +---+         +-----------+
 |   |         |           |
red red blue yellow blue yellow
         |           |
         +-----------+
```

The minimum height is 2 in this case.

The Input:

The first line of input gives the number of cases, *t* $(1 \le t \le 50)$. The *t* test cases follow. The first line of each case contains *n* $(1 \le n \le 500)$, the number of different colors for the marbles. The next line contains a string of *2n* words separated by spaces which correspond to the colors of the marbles, in order from left to right. Each color is a string of lower case letters ('a' .. 'z') no longer than 10 characters. There will be exactly *n* different colors and each color will appear exactly twice.

The Output:

For each test case, output one line containing the height of any optimal solution or -1 if no solution exists.

Sample Input:

```
4
3
red red blue yellow blue yellow
3
red blue yellow red blue yellow
3
red blue yellow blue yellow red
3
red red blue blue yellow yellow
```

Sample Output:

```
2
-1
3
1
```