Absolutely Sorted Array

Filename: `absolute`

When Atharva talks about numbers, he refers to a number as "bigger" than another if its absolute value is greater than the other number's absolute value. This definition causes confusion as -2 is bigger than 1, although it is less than 1, but it makes sense in his head. Atharva then gave Sharon an array of integers and asked him to sort them in nondecreasing order of "bigness", and if two numbers have the same "bigness", then compare them by their actual value. For example, the array [1, 0, 2, -2] would become [0, 1, -2, 2]. Sharon has no time to play Atharva's weird games so you will have to create a program to sort the array for him.

The Problem:

Given an array, sort it in nondecreasing order of absolute value, breaking ties by putting negative values at lower indices.

The Input:

The first line of the input file begins with a single, positive integer, t, representing the number of arrays. For each array, two lines follow. The first contains a single integer $1 \leq n \leq 100{,}000$ representing the number of arrays. The second line contains n integers $|a[i]| \leq 10^9$, representing the elements of the array.

The Output:

For each test case, output a single line containing "`Array #i: `" without the quotes, where `i` is the number of the array, followed by n space-separated integers, the elements of the sorted array. Make sure there are no trailing spaces.

Sample Input:

```
3
4
1 0 2 -2
7
1 -2 -3 4 5 6 -6
1
40
```

Sample Output:

```
Array #1: 0 1 -2 2
Array #2: 1 -2 -3 4 5 -6 6
Array #3: 40
```

Anderson, Please!

Filename: `anderson`

Anderson Fanderson is a goofy kid who always trips and falls. Every day, he buys a smoothie from Smoothie King and goes back home to work on flow problems. The problem is, Anderson trips every k steps he makes, spilling his some of his smoothie!

On his way home, Anderson will meet n people after some number of steps. If Anderson trips when he meets a person, he will spill his smoothie on them. Anderson always gets a large smoothie, so he will never run out of smoothie to spill.

The Problem:

Determine if Anderson will spill his smoothie on someone.

The Input:

The first line of the input file begins with a single, positive integer, t, representing the number of days. For each day, two lines follow. The first contains two integers $1 \leq n \leq 50$ and $1 \leq k \leq 10$, representing the number of people Anderson will meet on his journey home. The second line contains n integers $s_i \leq 100$, representing the number of steps Anderson takes before he will meet the ith person.

The Output:

For each test case, output a single line containing "`Day #i:`" without the quotes, where `i` is the number of the array, followed by a space and either "`Anderson, please!`" if he will spill the smoothie on at least one person, or "`Hey Andy!`" otherwise.

Sample Input:

```
3
5 1
1 2 3 4 5
3 2
3 5 6
3 3
10 7 2
```

Sample Output:

```
Day #1: Anderson, please!
Day #2: Anderson, please!
Day #3: Hey Andy!
```

Ant's Journey

Filename: `journey`

An ant lives on the cartesian plane. It has decided to go on a journey from the point (0, 0) to the point (n, m) by only making moves to the right or up. A move to the right increments the ant's x coordinate, while a move up increments the ant's y coordinate. However, there may be anteaters at several points along the way, which the ant must avoid. Determine how many different ways can the ant make its journey.

The Problem:

Given the location of anteaters, as well as the end point of the journey, count the number of ways in which the ant reach the end. As this number can be very large, print it modulo 10^9+7.

The Input:

The first line of the input file begins with a single, positive integer, t, representing the number of journeys. For each journey, several lines follow. The first contains two integers, $1 \leq n, m \leq 2000$, representing the journey's end point. The second line a single integer, $0 \leq k \leq 10000$ representing the number of anteaters. k lines follow, each with two integers, $0 \leq x, y \leq 2000$, representing the location of each anteater. No two anteaters share the same location.

The Output:

For each test case, output a single line saying "`Journey #i: c`" without the quotes, where `i` is the number of the journey, and `c` is the number of ways the ant can reach the endpoint modulo 10^9+7.

Sample Input:

```
3
3 4
3
0 2
2 4
1 3
1 1
0
10 10
2
9 5
5 9
```

Sample Output:

```
Journey #1: 14
Journey #2: 2
Journey #3: 160732
```

Find Parking

Filename: `parking`

Today Sally gave Ryan a ride to class. When they arrived at UCF, they could not decide where to park, so Ryan suggested that they just park at the closest parking garage. Parking garages at UCF are represented as rectangles with vertical and horizontal sides on the coordinate plane. The distance from Sally and Ryan to a parking garage is the shortest distance from their position to any point along the boundary of the parking garage.

The Problem:

Determine which parking garage is the closest to Sally and Ryan. It is guaranteed that one parking garage is closer than all the others.

The Input:

The first line of the input file begins with a single, positive integer, t, representing the number of days. For each day, multiple lines follow. The first line contains two integers, x and y, representing Sally and Ryan's current position. The second line contains a single integer, $1 \leq n \leq 26$, representing the number of parking garages at UCF. Then, n lines follow, each containing two pairs of integers, (x_1,y_1) and (x_2,y_2), such that $x_1 < x_2$ and $y_1 < y_2$, representing the bottom left and top right coordinates of the parking garage, respectively. No integer in the input has an absolute value greater than 10^6. It is guaranteed that Sally and Ryan are not currently inside a parking garage, and that parking garages do not intersect each other.

The Output:

For each test case, output a single line containing "`Day #i: Garage x`" without the quotes, where `i` is the number of the day, and `x` is a capital letter representing the number of the garage in the order given in the input (A comes first, then B, and so on).

Sample Input:

```
2
0 0
3
10 10 20 20
-20 -100 -10 100
0 1000 1 1001
999 0
3
10 10 20 20
-20 -100 -10 100
1000 0 1001 1
```

Sample Output:

```
Day #1: Garage B
Day #2: Garage C
```

Rap Song

Filename: `rap`

Sharon wants to write a rap song about the hardships of being a student, but he knows nothing about flow. We say a sentence flows if the first letter of every word is the same as the last letter of the word before it. If there is no word before it then the first letter can be anything. A word is a series of letters which is separated from other words by a space. For example, the sentences "students sometimes sleep poorly yo", "moms spaghetti", and "ba a ab" flow, but "ring ring banana phone" and "hello world dolphins" do not. Sharon wrote some lines for his new song, and now he wants you to determine if these lines flow.

The Problem:

Determine if each sentence flows.

The Input:

The first line of the input file begins with a single, positive integer, *t*, representing the number of lines written by Sharon. This is followed by *t* lines, each consisting of a string of no more than 100 lowercase letters and spaces.

The Output:

For each test case, output a single line containing "`Sentence #i:`" without the quotes, where `i` is the number of the sentence, followed by a space and either "`Flow`" if the sentence flows, or "`No go`" otherwise.

Sample Input:

```
3
moms spaghetti its serious stuff
im never gonna dance again
mamamia
```

Sample Output:

```
Sentence #1: Flow
Sentence #2: No go
Sentence #3: Flow
```

Room Temperature

Filename: `temperature`

Sharon built a new rectangular hotel with n x m rooms. When people move in, they each set the air conditioning (AC) to their preferred temperature setting, between 0 and 9. However, some rooms are unoccupied so the AC is not turned on, and the temperature in these rooms is unknown. What is known, however, is that the temperature in these rooms is equal to the average of the temperatures of all adjacent rooms. A room is adjacent to another if they are directly next to each other in the four cardinal directions (up, right, down, left).

For example, if a room is adjacent to three rooms with temperatures 3, 6, and 9, then the temperature in that room will be equal to (3+6+9)/3 = 6.

Sharon now wants to know what the temperature is in every room. It is guaranteed that at least one room is occupied, and therefore the temperature in that room is known. It is also guaranteed that at least one room is unoccupied.

The Problem:

Given information about each of the rooms in the hotel, determine the temperature in every unoccupied room.

The Input:

The first line of the input file begins with a single, positive integer, t, representing the number of hotels. For each hotel, multiple lines follow. The first contains two integers $1 \leq n, m \leq 10$ representing the length and width of the hotel. Then, n lines follow each with m characters, representing the n x m rooms. Each character is either a digit [0-9] representing the temperature setting, or the character '?' if the temperature is unknown.

The Output:

For each test case, output n+1 lines. The first line consists of a single string, "`Hotel #i:`" without the quotes, where `i` is the number of the hotel. The next n lines each contain m characters representing the temperature of every room. The temperatures in the output should be rounded to the nearest whole integer - so 3.50 becomes 4 and 3.49 becomes 3. Print a blank line in between each test case.

(Sample Input and Output are on the next page)

Sample Input:

```
3
2 3
1?1
449
3 3
9?0
???
2?3
4 4
????
?8??
???1
????
```

Sample Output:

```
Hotel #1:
121
449

Hotel #2:
940
542
233

Hotel #3:
7765
7854
6641
6542
```

Note: There is a blank line after the last test case.

War of the Giants

Filename: `war`

The world is thrown into disarray as its two most powerful countries, Aliland and Buzzland (which we will refer to as giants), have gone to war. Unfortunately, all other countries are stuck in the middle of these two giants. These countries are forced to make a choice in the interest of their security - they must ally either with Aliland or Buzzland.

Each country will choose to ally with the closer of the two giants. A giant is closer than another if the shortest path between the country and the giant is shorter than the other. If there is no path between the countries then the length of the shortest path is infinity. If both giants are equally close, the country will remain neutral.

For each country, determine its ally or whether it remains neutral. Aliland is represented as country 1 and Buzzland is represented as country 2.

The Problem:

Given a graph where each node represents a country, determine the alliances formed in the War of the Giants.

The Input:

The first line of the input file begins with a single, positive integer, t, representing the number of war. For each wars, two lines follow. The first contains a two integers $3 \leq n$, $m \leq 100{,}000$ representing the number of countries and the number of roads. m lines follow, each with three integers, $1 \leq u$, $v \leq n$ and $1 \leq w \leq 10^9$, representing a bidirectional road between countries u and v with length w. There will be no duplicate edges.

The Output:

For each test case, output a single line saying "`War #i: c`" without the quotes, where `i` is the number of the war, and `c` is a string of n characters, either 'A', 'B', or 'N' at the jth position depending on whether the jth country allies with Aliland, Buzzland, or remains neutral. Aliland and Buzzland are allied to themselves by default.

(Sample Input and Output are on the next page)

Sample Input:

```
2
5 7
1 3 10
2 3 10
1 4 5
4 5 5
2 5 5
3 4 2
3 5 7
7 7
1 2 1
1 7 1
2 3 1
3 4 1
4 5 1
5 6 1
6 7 1
```

Sample Output:

```
War #1: ABAAB
War #2: ABBBNAA
```