Problem 1 (50 points) Dam Breaking

Save the village from flooding! The dam is weakening! You are the only person with the skills to fix cracks in the dam. Cracks of different sizes appear at different times. Every time unit they remain unfixed, they get a little larger. You can work fast enough to fix one crack every time unit. Each time unit, the ground in the village can absorb some of the incoming water, but if the total floodwaters reach or exceed a certain level, the village must be evacuated.

Problem 1 (50 points) Dam Breaking

Input specification: the first line contains the integer n, representing the number of cracks that occur in the dam.

The second line contains a positive integer indicating the threshold (units of water) for how much flooding the village can withstand before having to evacuate. If the floodwaters equal or exceed this threshold, the village is considered flooded and must be evacuated.

The third line contains a positive integer indicating the amount (units of water) that drain out of the village each time unit.

Following that are n lines. Each line contains 2 values, separated by a space, that specify a particular crack. The first value on a line indicates the time unit that the crack appears. This value is guaranteed to be a non-negative integer. The second value on the line indicates the initial size associated with this crack. This value is guaranteed to be a positive integer. It represents the number of units of water that can flow through it in one time unit. You may assume that n is not bigger than 1,000,000 and that each input value fits in a 32-bit integer. You may also assume that the input file lists the cracks in non-decreasing order of time of appearance.

Problem 1 (50 points) Dam Breaking

You are able to fix one crack per time unit, and can choose to fix a crack in the same time unit that it appears, or delay it for a later time unit. Note that there may be multiple cracks that appear at the same time unit. (The problem is trivial if this doesn't occur.) At the end of a time unit, all remaining,

unfixed cracks increase in size by one.

To simplify, we will discretize these continuous processes as follows. In time unit t:

- All new cracks that appear in time unit t appear at the beginning of the time unit.
- You are able to instantaneously fix one crack that appeared at time unit t, or that appeared earlier but was not fixed yet.
- The floodwaters in the village are adjusted in a single calculation, that adds floodwater according to the total size of all the current, unfixed
- cracks, and decreases floodwater according to the amount that can be drained in one time unit. The floodwaters can never go below zero (you
- can't drain water that isn't present). If the floodwaters ever equal or exceed the threshold, the village is flooded and must be evacuated at that
- point.
- Finally, all of the remaining cracks increase in size by 1. This essentially happens all at once as the current time unit ends.

Problem 1 (50 points) Dam Breaking

Output specification: to help with debugging, the output includes a couple of pieces of information. If flooding occurs, the output contains three lines

of information. The first line is the word "FLOOD". The second line contains an integer indicating which time unit resulted in flooding. The third line

indicates the total units of floodwater present when flooding occurred. If flooding does not occur, the output contains two lines of information. The

first line is the word "SAFE". The second line contains an integer indicating the maximum value the floodwaters ever reached.

You should submit a readme.txt file with an explanation of your code and algorithms. You must provide exact instructions on how to run your code and you must submit screen shots of your running code on all provided inputs and printing your result to the console

Problem 1 (50 points) Dam Breaking

Sample input file:

```
2
10
3
0 7
0 8
```

Output:

```
python3 flood.py < flood_inputs/flood_1.txt
SAFE
4
```

Problem 1 (50 points) Dam Breaking

Sample input file:

```
3
11
2
0 7
0 8
1 9
```

Output:

```
python3 flood.py < flood_inputs/flood_2.txt
FLOOD
1
11
```

Problem 2 (50 points) Downhill Skier

You are a downhill skier looking for an adventure. You will be dropped by helicopter as a location of your choosing on a mountain. The mountain is represented as an m x n matric of altitude values. Design an algorithm that determines the longest path you can follow down the mountain such that each successive cell in your path has an altitude lower than the previous cell.

Note: Each Cell has up to eight neighbors, directly adjacent horizontally, vertically or diagonally.

Output the longest path to the terminal.

You should submit a readme.txt file with an explanation of your code and algorithms. You must provide exact instructions on how to run your code and you must submit screen shots of your running code on all provided inputs and printing your result to the console.

Problem 2 (50 points) Downhill Skier

Sample input:

```
4
5
4 8 7 9 8
6 5 10 12 11
3 6 15 11 12
1 4 9 13 10
```

```
python3 ski.py < ski-inputs/ski_input1.txt
9
```