ACM South American Regional

Collegiate Programming Contest

Warm-up Session

November 10, 2001

Buenos Aires – Argentina
Natal – Brazil
Caxias do Sul – Brazil
Campinas – Brazil
Punta Arenas – Chile
Margarita – Venezuela

PROBLEM 1: Rainy Flatland

File Names

       Source File: rain.c, rain.pas, etc...
       Input File: rain.in
       Output File: rain.out

Statement of the Problem

In a plane area, which suffers from chronic flood problems, the inhabitants decided to dig a series of wells, in order to collect rainwater. The excavation of a well is done in steps, so that the workers can go up and down in the well. According to the norms of the Union of Well Diggers (UWD), the mouth of the well must have a rectangular shape, with length and width never exceeding 50 meters and always integer quantities. After a certain depth, between 1 and 10 meters, a step is created, thereby reducing the width and length of the current hole by one meter. The excavation continues until the UWD decides to stop it or until it is impossible to continue because one of the dimensions would be reduced to zero. The walls of the wells are perfectly vertical, a fact that the UWD is proud of.

    The inhabitants would like to known if the number of wells is enough to drain the water or not. Your team was hired to verify the extent to which the wells fill, given a certain amount of rain (in cubic meters). It should be noted that the wells are interlinked by an intricate net of pipes and water registers, in such way that initially the first well fills totally, then the second well begins to be filled, and so on. As this net of pipes is only known to the UWD, it is not described here, and you should assume that it is irrelevant for the problem. But it works, and the wells are filled with rain water in exactly the same order that they are described in the input file.

Input Format

The input file consists of several test cases, each describing a series of wells and the expected rainfall. Each test case begins with a line containing a number n indicating the number of wells $(0 < n < 60)$. This line is followed by n groups of lines describing well 1, well 2, up to well n. The first line in the group contains the length and width of the well's mouth , separated by one or more spaces. The next line contains a number m, the number of steps of that well. Then m lines follow, each giving the depth of a step, always measured in relation to the ground level (not in relation to the bottom of the well, nor in relation with the other steps); the last depth is actually the bottom of that well. The numbers are given in ascending order, from the first step (nearest to the ground) to the last one (deepest). All measurements are in meters, and always integer numbers (a tradition of the UWD). After the well descriptions, a line with an integer number indicates the expected amount of rain, in cubic meters. After that, a new test case begins. The case $n = 0$ indicates the end of the test cases.

Output Format

For the ith test case of the input file, the line
    Test #i: water level at well j is at x meters
should be printed, where j is the first well which is not completely filled, and x is the water level in well j (measured from the mouth of the well, i.e, the ground level, and rounded (down) to two

decimal places). If the wells are completely filled with the rainwater, this should be informed with the sentence

    Test #i: it will flood by x cubic meters.

Note that x could be zero in the case that the rainfall has exactly the same volume as the wells.

Sample Input

```
2
5 10
3
3
8
17
2 2
1
6
500
1
1 1
1
5
30
0
```

Sample Output

```
Test  #1: water level at well  1 is at 0.92 meters
Test  #2: it will flood by 25 cubic meters
```

Note that if the water amount in the first example were 550 cubic meters, the answer would be

    Test #1: water level at well 2 is at 5.00 meters

PROBLEM 2: Bob and his bike

File Names

> Source File: bike.c, bike.pas, etc
> Input File: bike.in
> Output File: bike.out

Statement of the Problem

Bob has received medical advice to practice sports daily. Since his job is not very far from home, he decided to bike to work using the wonderful bikeway system of his town. At the end of the day, since he is tired, he goes back home by train, carrying his bike. Since he has not done any physical activity for a while, his doctor recommended a gradual training program, with a prescription of the number of daily kilometers to be conquered in every week of the training program.

The bikeways at Bob's town are very peculiar: they are all 1km long and are one way only. Moreover, the city has many parks connected by such bikeways and some of these parks are connected (through bikeways) to Bob's neighborhood and to the building where Bob works. Bob then decided to follow the training program by visiting an increasing number of parks each week, on his way to work.

Your work is to write a computer program that receives as input the training program prepared by Bob's doctor, along with a description of the bikeway system, and decides if the program can be fulfilled.

Input Format

The input file may contain several instances of the problem, with the description of a training program and a bikeway system. Each instance is described as follows:

1. A line with the number of weeks, n, of the training program. A program with $n = 0$ marks the end of the input;

2. a line with n numbers, with the number of kilometers to be covered daily in every week;

3. a line with the number of bikeways in Bob's town;

4. a list of pairs describing the origin/destination of each bikeway. Letter H represents Bob's home, letter J represents Bob's work, and P1,...,PN represent the N parks in town.

In each line, the elements may be separated by an arbitrary number of spaces.

Output Format

For the ith program in the input file, the ith line in the output file must contain the words `Possible program`, if the corresponding program can be followed, or `Impossible program`, otherwise.

3

Sample Input

```
10
1 2 3 4 5 6 7 9 34 98
5
H  P1
P1 P2
P2 P3
J  P3
P2 J
3
1 2 3
6
H  P1
H  P2
H  J
P1 J
P2 J
P2 P1
5
2 4 6 8 10
4
H  P1
P1 P2
P2 P1
P1 J
5
2 4 6 9
4
H  P1
P1 P2
P2 P1
P1 J
0
```

Sample Output

```
Impossible program
Possible program
Possible program
Impossible program
```