TU-Darmstadt Programming Contest

Team Contest 2008

June 21, 2008

PROBLEM SET

This problem set should contain 8 (eight) problems on 11 (eleven) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Contents

General Remark

- All programs read their input from `stdin` and output to `stdout` (no files allowed). `stderr` can be used for test outputs. You can rely on correct input specifications (no error handling code on input required).
- In Java solutions, the `main` method has to be in a public class named `Solution`.
- If not specified otherwise, all numbers can be assumed to fit into a 32-bit signed integer.

Have fun!

1 Dart Challenge

Clark and Harry are siblings. As they had been rivals since their early childhood, their father decided that both should concentrate on a different sport when they were thirteen. That way, they would not have to compete for success. Now both are twenty years old and excel in different fields: Clark plays chess while Harry participates in dart-tournaments.

Having won a series of three tournaments in a row, Harry started teasing Clark about not having as much success. Clark retorted that chess was less luck-based and thus more difficult. That offended Harry and led him to the reply that in order to play darts optimally, a lot of combinatorics are necessary. Clark returned an icy smile and the comment that memorizing all different late-games could hardly be called "combinatorics".

This is how it came to the wager. Harry bets that he can find all possible late-games for generalized dart-boards where memorized late-games do not help him. When Clark showed him a list of possible dart-boards, Harry had to admit that he probably bit off more than he can chew. As his friend, you have to help him!

Problem

A dart-board consists of different areas. Each area has an assigned score for hitting it. Each area also has a double- and a triple-field that are worth twice and three times the score of the area. The only exception is the area for the highest score: It has only a double- and *no triple*-field! Given the values of the different areas you have to find the number of possible scores that can be obtained with a given number of darts.

Input

The inputs start with a line containing a single integer n. Each of the n following lines contains one test case. Each test case starts with two integers $1 \leq a \leq 100, 1 \leq k \leq 50$, the number of different areas on the dart-board and the number of darts. a integers $1 \leq s_i \leq 100$ follow. s_i is the score for hitting area i. All scores are distinct. Remember that each area has a double- and, with exception of the area with the highest score, a triple-field. It is always possible to score 0 with any given dart by not hitting the board.

Output

The output for every test case begins with a line containing "`Scenario #i:`", where `i` is the number of the scenario counting from 1. After that, output a single line containing the number of different scores that can be obtained with k darts on the given board. Terminate each test case with an empty line.

Sample Input

```
3
21 3 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 25
2 2 20 10
1 50 1
```

Sample Output

```
Scenario #1:
172

Scenario #2:
9

Scenario #3:
101
```

2 Online Shopping

The Internet becomes more and more important for our daily life. Aside from information retrieval, many people use the web for comfortable shopping from their home PCs. As the number of online customers grows, so does the number of websites dedicated to comparing online prices. Competitive sites need to quickly visualize the cheapest offers for a certain product. Barter & Haggle Inc. has recently been successful in the field of comparing online prices. However, as more and more comparable services appear, B&H has trouble conserving their market share. This is why the company has decided to improve its technology by comparing prices for different online shops and products simultaneously. The engineers have not been capable of implementing the visualization algorithm, though. This is where you come into play.

Problem

Given a table of prices for different products at different online stores, you need to find an optimal reordering of the rows and columns. In order to compare different reorderings, we define the *table string* to be the string of cells that is obtained from a table by appending all columns. An optimal reordering has the smallest table string with respect to lexicographical comparison (the table string is compared cell-wise).

	EasySwush	Sucker	DustDestroyer 5P
www.cleanup.com	39.99	40.00	129.99
www.shiny.net	50.00	40.00	99.99

39.99
50.00
40.00
40.00
129.99
99.99

Table 1: Optimal table of prices for vacuum cleaners and table string.

Input

The inputs start with a line containing a single integer n. Each of the n following lines contains one test case. Each test case starts with two integers $1 \leq a, b \leq 5$, the number of products and online shops respectively. The table string consisting of the prices separated by single spaces follows. Each price p is given as an integer amount of cents with $0 \leq p \leq 10^9$.

Output

The output for every test case begins with a line containing "`Scenario #i:`", where `i` is the number of the test case counting from 1. Then, output a single line containing the table string of the optimal reordering of products and shops. Terminate each test case with an empty line.

Sample Input

```
2
3 2 3999 5000 4000 4000 12999 9999
4 3 120 120 110 120 80 75 250 50 200 55 80 80
```

Sample Output

```
Scenario #1:
3999 5000 4000 4000 12999 9999

Scenario #2:
50 200 250 80 75 120 80 80 55 120 110 120
```

3 Playground Hideout

Little Timmy likes playgrounds, especially those with complicated constructions of wooden towers interconnected with plank bridges or ropes, with slides and rope-ladders. He could play on his favourite playground for days, if only his parents let him. Sooner or later they decide that it is time to go home. For their next trip to a playground Timmy has a plan: he will not simply let his father grab him, but climb to the highest platform of the most complex structure and hide there. His father will never be able to reach him there or at least it will give Timmy some extra time.

Problem

An adventure playground consists of several platforms. The difficulty of reaching a platform directly from ground level varies. In addition the different platforms are interconnected by "bridges" of different difficulty. There are connections that can be used a lot more easily in one direction than in the other, e.g. slides. Given a plan of an adventure playground you need to help Timmy find the platform that is most difficult to reach from ground level. The difficulty for a path in the adventure playground can be estimated as a sum of the difficulties of the connections used. The difficulty of reaching a platform is the difficulty of the least difficult path from ground level to the platform.

Input

The input starts with a line containing a single integer n, the number of test cases. Each test case starts with a line containing two integers $1 \leq p, c \leq 10000$, the number of platforms and connections respectively (we allow for large playgrounds). A line with p integers $p_i \in [0, 1000]$ follows, where p_i is the difficulty of reaching platform i directly from ground-level. The next c lines each describe a connection between two platforms. They consist of four integers i, j, a, b; $i < j$, the zero-based index of the two connected platforms and the difficulties $a \in [0, 1000]$ and $b \in [0, 1000]$ of using the connection to get from p_i to p_j and from p_j to p_i respectively. There can be multiple connections between any two platforms.

Output

The output for every test case begins with a line containing "`Scenario #i:`", where `i` is the number of the test case counting from 1. Then, output a single line containing the zero-based index of the platform that is most difficult to reach from ground level. If two or more platforms are equally difficult to reach, output the smallest index of among those platforms. Terminate each test case with an empty line.

Sample Input

```
2
4 3
1 10 10 40
0 3 3 2
1 3 4 3
2 3 5 4
2 2
11 10
0 1 2 2
0 1 1 0
```

Sample Output

```
Scenario #1:
2

Scenario #2:
0
```

4 Rain on your Parade

Background

You're giving a party in the garden of your villa by the sea. The party is a huge success, and everyone is here. It's a warm, sunny evening, and a soothing wind sends fresh, salty air from the sea. The evening is progressing just as you had imagined. It could be the perfect end of a beautiful day.

But nothing ever is perfect. One of your guests works in weather forecasting. He suddenly yells, "I know that breeze! It means its going to rain heavily in just a few minutes!" Your guests all wear their best dresses and really would not like to get wet, hence they stand terrified when hearing the bad news.

You have prepared a few umbrellas which can protect a few of your guests. The umbrellas are small, and since your guests are all slightly snobbish, no guest will share an umbrella with other guests. The umbrellas are spread across your (gigantic) garden, just like your guests. To complicate matters even more, some of your guests can't run as fast as the others.

Can you help your guests so that as many as possible find an umbrella before it starts to pour?

Problem

Given the positions and speeds of all your guests, the positions of the umbrellas, and the time until it starts to rain, find out how many of your guests can at most reach an umbrella. Two guests do not want to share an umbrella, however.

Input

The input starts with a line containing a single integer, the number of test cases.

Each test case starts with a line containing the time t in minutes until it will start to rain ($1 \leq t \leq 5$). The next line contains the number of guests m ($1 \leq m \leq 3000$), followed by m lines containing x- and y-coordinates as well as the speed s_i in units per minute ($1 \leq s_i \leq 3000$) of the guest as integers, separated by spaces. After the guests, a single line contains n ($1 \leq n \leq 3000$), the number of umbrellas, followed by n lines containing the integer coordinates of each umbrella, separated by a space.

The absolute value of all coordinates is less than 10000.

Output

For each test case, write a line containing "`Scenario #i:`", where `i` is the number of the test case starting at 1. Then, write a single line that contains the number of guests that can at most reach an umbrella before it starts to rain. Terminate every test case with a blank line.

Sample Input

```
2
1
2
1 0 3
3 0 3
2
4 0
6 0
1
2
1 1 2
3 3 2
2
2 2
4 4
```

Sample Output

```
Scenario #1:
2

Scenario #2:
2
```

5 Olympic Games

Mike loves the olympic games. What he hates about olympic games – besides them taking place only once every four years – is that the individual events are scheduled concurrently. That way it is never possible to watch all competitions live. Mike always tries to plan his personal schedule in order to maximize the number of events he can attend to. However, at the end he is never sure whether or not his schedule actually is optimal. Help him!

Problem

Given the days and times on which events take place, determine the maximum number of events Mike can watch. Mike never leaves during an event, so it is not possible to partially attend to events.

Input

The input starts with a line containing a single integer n, the number of test cases.

Each test case starts with a line containing an integer $1 \leq m \leq 50000$ that specifies the number of different events. The next m lines describe one event with three integers d, s, e. The event takes place on day d of the olympics, it starts at s and it ends at e, where the times are given in the form *hhmm*. All events are assumed to conclude the same day they started.

Output

The output for every test case begins with a line containing "`Scenario #i:`", where `i` is the number of the test case counting from 1. Then, output a single line containing the number of events Mike can attend to at most. The time for getting from one event to another can be neglected. Events starting at time t can be scheduled after events ending at t as long as there are no other conflicts. Terminate each test case with an empty line.

Sample Input

```
2
10
1 1220 1340
2 1155 1220
2 1220 1340
3 1220 1240
1 1200 1320
2 1250 1310
2 1330 1550
3 1030 1130
3 1130 1300
3 1240 1330
3
1 0500 2200
1 0000 0700
1 2000 2359
```

Sample Output

```
Scenario #1:
7

Scenario #2:
2
```

6 Filthy Rich

They say that in Phrygia, the streets are paved with gold. You're currently on vacation in Phrygia, and to your astonishment you discover that this is to be taken literally: small heaps of gold are distributed throughout the city. On a certain day, the Phrygians even allow all the tourists to collect as much gold as they can in a limited rectangular area. As it happens, this day is tomorrow, and you decide to become filthy rich on this day. All the other tourists decided the same however, so it's going to get crowded. Thus, you only have one chance to cross the field. What is the best way to do so?

Problem

Given a rectangular map and amounts of gold on every field, determine the maximum amount of gold you can collect when starting in the upper left corner of the map and moving to the adjacent field in the east, south, or south-east in each step, until you end up in the lower right corner.

Input

The input starts with a line containing a single integer, the number of test cases.

Each test case starts with a line, containing the two integers r and c, separated by a space ($1 \leq r, c \leq 1000$). This line is followed by r rows, each containing c many integers, separated by a space. These integers tell you how much gold is on each field. The amount of gold never negative.

The maximum amount of gold will always fit in an `int`.

Output

For each test case, write a line containing "`Scenario #i:`", where `i` is the number of the test case, followed by a line containing the maximum amount of gold you can collect in this test case. Finish each test case with an empty line.

Sample Input

```
1
3 4
1 10  8  8
0  0  1  8
0 27  0  4
```

Sample Output

```
Scenario #1:
42
```

7 Robotic Invasion

The pacifistic people of planet Pax are at war with the evil guys from planet Googol. Although they are strictly pacifistic they have means of defense. Their cryptographers are able to routinely decrypt the commands sent to Googol's robots and they can – using huge amounts of energy – construct bogus commands. It is the job of the Tactical Unit Defense (TUD) to find the best way to interfere with the command transmission from Googol.

Problem

You are given a string of movements (each being a step to the north, east, south or west) and a map containing an invading robot from Googol and, possibly, several obstacles and traps.

Your task is to find a way to overcome the threat by changing as few movements as possible and guiding the robot into a trap. A single change replaces a step by another direction or no movement at all.

A robot will move in the given direction if the position is not blocked or outside the given map. If given such a direction, the robot does not move instead.

Input

The first line contains the number of scenarios. For each scenario a line containing two integers $1 \leq w, h \leq 100$ separated by a single space is given, followed by h lines each containing w characters. This is a map of the area.

Exactly one character is a `R` which is the starting position of the invading robot. Other characters are '`.`' to denote free positions, `X` to denote blocked positions and + to denote traps.

This is followed by a line containing an integer $0 \leq c \leq 100$ and another line containing c characters of communication. Each character is either `N`, `E`, `S` or `W` to denote a command to move one position up, right, down or left, respectively.

Output

The output for every scenario begins with a line containing "`Scenario #i:`", where `i` is the number of the scenario starting at 1.

Then print on a single line the string of movements (denoted by characters `N`, `E`, `S`, `W` and, for no movement, `X`) that leads into a trap, of those the one that changes as few characters as possible, of those the one that leads into a trap as early as possible, of those the lexicographically first. This string must have c characters.

If no such string exists, output "`impossible`" on a single line. Terminate the output for the scenario with a blank line.

Sample Input

```
3
3 2
R+.
...
3
EEE
3 3
R..
.X+
...
3
SSE
3 3
R..
..+
...
3
SSE
```

Sample Output

```
Scenario #1:
EEE

Scenario #2:
EES

Scenario #3:
ESE
```

8 Higher Math

You are building a house. You'd prefer if all the walls have a precise right angle relative to the ground, but you have no device to measure angles. A friend says he has a great idea how you could ensure that all walls are upright: All you need to do is step away a few feet from the wall, measure how far away you are from the wall, measure the height of the wall, and the distance from the upper edge of the wall to where you stand. You friend tells you to do these measurements for all walls, then he'll tell you how to proceed. Sadly, just as you are done, a timber falls on your friend, and an ambulance brings him to the hospital. This is too bad, because now you have to figure out what to do with your measurements yourself.

Problem

Given the three sides of a triangle, determine if the triangle is a right triangle, i.e. if one of the triangle's angles is 90 degrees.

Input

The inputs start with a line containing a single integer n. Each of the n following lines contains one test case. Each test case consists of three integers $1 \leq a, b, c \leq 40000$ separated by a space. The three integers are the lengths of the sides of a triangle.

Output

The output for every scenario begins with a line containing "`Scenario #i:`", where `i` is the number of the scenario counting from 1. After that, output a single line containing either the string "yes" or the string "no", depending on if the triangle in this test case has a right angle. Terminate each test case with an empty line.

Sample Input

```
2
36 77 85
40 55 69
```

Sample Output

```
Scenario #1:
yes

Scenario #2:
no
```