Campus Test
Powered by Google Code Jam

Practice Round China New Grad Test 2014

A. Bad Horse

B. Captain Hammer

C. Moist

Questions asked

― Top Scores

yefllowers	65
akaring	65
AlphardWang	65
Prowindy	65
levy0834	65
yuxuelu9	65
Konjac	65
TimShen	65
LTzycLT	65
lxc902	65

Problem A. **Bad Horse**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 1
12 points   [Solve A-small-1]

Small input 2
21 points   [Solve A-small-2]

Problem

As the leader of the Evil League of Evil, Bad Horse has a lot of problems to deal with. Most recently, there have been far too many arguments and far too much backstabbing in the League, so much so that Bad Horse has decided to split the league into two departments in order to separate troublesome members. Being the Thoroughbred of Sin, Bad Horse isn't about to spend his valuable time figuring out how to split the League members by himself. That what he's got you -- his loyal henchman -- for.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a positive integer **M** on a line by itself -- the number of troublesome pairs of League members. The next **M** lines each contain a pair of names, separated by a single space.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is either "Yes" or "No", depending on whether the League members mentioned in the input can be split into two groups with neither of the groups containing a troublesome pair.

Limits

1 ≤ **T** ≤ 100.
Each member name will consist of only letters and the underscore character.
Names are case-sensitive.
No pair will appear more than once in the same test case.
Each pair will contain two distinct League members.

Small dataset

1 ≤ **M** ≤ 10.

Large dataset

1 ≤ **M** ≤ 100.

Sample

```
Input                                         Output

2                                             Case #1: Yes
1                                             Case #2: No
Dead_Bowie Fake_Thomas_Jefferson
3
Dead_Bowie Fake_Thomas_Jefferson
Fake_Thomas_Jefferson Fury_Leika
Fury_Leika Dead_Bowie
```

Powered by

Google Cloud Platform

Campus Test
Powered by Google Code Jam

Questions asked

⊟ Submissions

Bad Horse

| 12pt | Not attempted **789/1984 users** correct (40%) |
| 21pt | Not attempted **707/776 users** correct (91%) |

Captain Hammer

| 22pt | Not attempted **755/1070 users** correct (71%) |

Moist

| 4pt | Not attempted **793/1049 users** correct (76%) |
| 6pt | Not attempted **770/775 users** correct (99%) |

⊟ Top Scores

yefllowers	65
akaring	65
AlphardWang	65
Prowindy	65
levy0834	65
yuxuelu9	65
Konjac	65
TimShen	65
LTzycLT	65
lxc902	65

Problem B. Captain Hammer

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

| Small input 22 points | Solve B-small |

Problem

The Hamjet is a true marvel of aircraft engineering. It is a jet airplane with a single engine so powerful that it burns all of its fuel instantly during takeoff. The Hamjet doesn't have any wings because who needs them when the fuselage is made of a special Wonderflonium isotope that makes it impervious to harm.

Piloting the Hamjet is a not a job for your typical, meek-bodied superhero. That's why the Hamjet belongs to Captain Hammer, who is himself impervious to harm. The G-forces that the pilot endures when taking a trip in the Hamjet are legen-dary.

The Hamjet takes off at an angle of θ degrees up and a speed of **V** meters per second. **V** is a fixed value that is determined by the awesome power of the Hamjet engine and the capacity of its fuel tank. The destination is **D** meters away. Your job is to program the Hamjet's computer to calculate θ given **V** and **D**.

Fortunately, the Hamjet's Wondeflonium hull is impervious to air friction. Even more fortunately, the Hamjet doesn't fly too far or too high, so you can assume that the Earth is flat, and that the acceleration due to gravity is a constant 9.8 m/s^2 down.

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line will contain two positive integers -- **V** and **D**.

Output

For each test case, output one line containing "Case #x: θ", where x is the case number (starting from 1) and θ is in degrees up from the the horizontal. If there are several possible answers, output the smallest positive one.

An answer will be considered correct if it is within 10^{-6} of the exact answer, in absolute or relative error. See the FAQ for an explanation of what that means, and what formats of floating-point numbers we accept.

Limits

1 ≤ **T** ≤ 4500;
1 ≤ **V** ≤ 300;
1 ≤ **D** ≤ 10000;
It is guaranteed that each test case will be solvable.

Sample

```
Input          Output

3              Case #1: 45.0000000
98 980         Case #2: 15.0000000
98 490         Case #3: 3.8870928
299 1234
```

Problem C. Moist

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 1 4 points	Solve C-small-1
Small input 2 6 points	Solve C-small-2

Problem

Moist has a hobby -- collecting figure skating trading cards. His card collection has been growing, and it is now too large to keep in one disorganized pile. Moist needs to sort the cards in alphabetical order, so that he can find the cards that he wants on short notice whenever it is necessary.

The problem is -- Moist can't actually pick up the cards because they keep sliding out his hands, and the sweat causes permanent damage. Some of the cards are rather expensive, mind you. To facilitate the sorting, Moist has convinced Dr. Horrible to build him a sorting robot. However, in his rather horrible style, Dr. Horrible has decided to make the sorting robot charge Moist a fee of $1 whenever it has to move a trading card during the sorting process.

Moist has figured out that the robot's sorting mechanism is very primitive. It scans the deck of cards from top to bottom. Whenever it finds a card that is lexicographically smaller than the previous card, it moves that card to its correct place in the stack above. This operation costs $1, and the robot resumes scanning down towards the bottom of the deck, moving cards one by one until the entire deck is sorted in lexicographical order from top to bottom.

As wet luck would have it, Moist is almost broke, but keeping his trading cards in order is the only remaining joy in his miserable life. He needs to know how much it would cost him to use the robot to sort his deck of cards.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each one starts with a line containing a single integer, **N**. The next **N** lines each contain the name of a figure skater, in order from the top of the deck to the bottom.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the number of dollars it would cost Moist to use the robot to sort his deck of trading cards.

Limits

1 ≤ **T** ≤ 100.
Each name will consist of only letters and the space character.
Each name will contain at most 100 characters.
No name with start or end with a space.
No name will appear more than once in the same test case.
Lexicographically, the space character comes first, then come the upper case letters, then the lower case letters.

Small dataset

1 ≤ **N** ≤ 10.

Large dataset

1 ≤ **N** ≤ 100.

Sample

Input	Output
2 2 Oksana Baiul Michelle Kwan 3 Elvis Stojko Evgeni Plushenko Kristi Yamaguchi	Case #1: 1 Case #2: 0

Campus Test
Powered by Google Code Jam

Questions asked

▬ Submissions

Read Phone Number

6pt	Not attempted **1885/3058 users** correct (62%)
13pt	Not attempted **1094/1837 users** correct (60%)

Rational Number Tree

9pt	Not attempted **1193/1545 users** correct (77%)
12pt	Not attempted **368/1037 users** correct (35%)

Sorting

5pt	Not attempted **1666/1990 users** correct (84%)
8pt	Not attempted **1551/1635 users** correct (95%)

Cross the maze

10pt	Not attempted **134/370 users** correct (36%)
13pt	Not attempted **119/132 users** correct (90%)

Spaceship Defence

10pt	Not attempted **175/382 users** correct (46%)
14pt	Not attempted **106/152 users** correct (70%)

▬ Top Scores

dreamoon	100
springegg	100
tckwok	100
cgy4ever	100
OR.Director	100
AlanC	100
Mochavic	100
jxwuyi	100
oldherl	100
Descent	100

Problem A. Read Phone Number

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 6 points	Solve A-small
Large input 13 points	Solve A-large

Problem

Do you know how to read the phone numbers in English? Now let me tell you.

For example, In China, the phone numbers are 11 digits, like: 15012233444. Someone divides the numbers into 3-4-4 format, i.e. 150 1223 3444. While someone divides the numbers into 3-3-5 format, i.e. 150 122 33444. Different formats lead to different ways to read these numbers:

150 1223 3444 reads one five zero one double two three three triple four.

150 122 33444 reads one five zero one double two double three triple four.

Here comes the problem:

Given a list of phone numbers and the dividing formats, output the right ways to read these numbers.

Rules:

Single numbers just read them separately.

2 successive numbers use double.

3 successive numbers use triple.

4 successive numbers use quadruple.

5 successive numbers use quintuple.

6 successive numbers use sextuple.

7 successive numbers use septuple.

8 successive numbers use octuple.

9 successive numbers use nonuple.

10 successive numbers use decuple.

More than 10 successive numbers read them all separately.

Input

The first line of the input gives the number of test cases, **T**. **T** lines|test cases follow. Each line contains a phone number **N** and the dividing format **F**, one or more positive integers separated by dashes (-), without leading zeros and whose sum always equals the number of digits in the phone number.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the reading sentence in English whose words are separated by a space.

Limits

$1 \le$ **T** ≤ 100.

Small dataset

$1 \le$ **length of N** ≤ 10.

Large dataset

$1 \le$ **length of N** ≤ 100.

Sample

```
Input

3
15012233444 3-4-4
15012233444 3-3-5
```

```
12223 2-3
```

Output

```
Case #1: one five zero one double two three three triple f
Case #2: one five zero one double two double three triple
Case #3: one two double two three
```

Powered by

Google Cloud Platform

Campus Test
Powered by Google Code Jam

Problem B. **Rational Number Tree**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

| Small input 9 points | Solve B-small |
| Large input 12 points | Solve B-large |

Problem

Consider an infinite complete binary tree where the root node is 1/1 and left and right childs of node p/q are p/(p+q) and (p+q)/q, respectively. This tree looks like:

```
            1/1
      _____|_____
     |             |
    1/2           2/1
   __|__         __|__
  |     |       |     |
 1/3   3/2     2/3   3/1
...
```
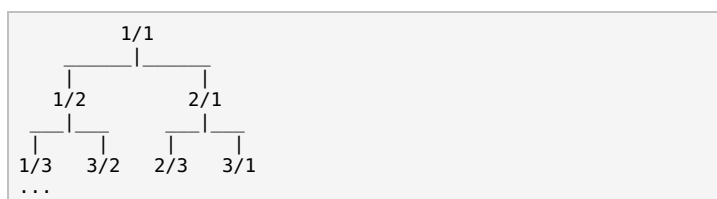
It is known that every positive rational number appears exactly once in this tree. A level-order traversal of the tree results in the following array:

```
1/1, 1/2, 2/1, 1/3, 3/2, 2/3, 3/1, ...
```

Please solve the following two questions:

1. Find the **n**-th element of the array, where **n** starts from 1. For example, for the input 2, the correct output is 1/2.
2. Given **p/q**, find its position in the array. As an example, the input 1/2 results in the output 2.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line. The line contains a problem id (1 or 2) and one or two additional integers:

1. If the problem id is 1, then only one integer **n** is given, and you are expected to find the **n**-th element of the array.
2. If the problem id is 2, then two integers **p** and **q** are given, and you are expected to find the position of **p/q** in the array.

Output

For each test case:

1. If the problem id is 1, then output one line containing "Case #x: p q", where x is the case number (starting from 1), and p, q are numerator and denominator of the asked array element, respectively.
2. If the problem id is 2, then output one line containing "Case #x: n", where x is the case number (starting from 1), and n is the position of the given number.

Limits

$1 \le T \le 100$; **p** and **q** are relatively prime.

Small dataset

$1 \le n, p, q \le 2^{16}-1$; **p/q** is an element in a tree with level number ≤ 16.

Large dataset

$1 \le n, p, q \le 2^{64}-1$; **p/q** is an element in a tree with level number ≤ 64.

Sample

Input	Output
4	Case #1: 1 2
1 2	Case #2: 2

```
2 1 2     Case #3: 3 2
1 5       Case #4: 5
2 3 2
```

Problem C. Sorting

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the **Quick-Start Guide** to get started.

Small input	[Solve C-small]
5 points	
Large input	[Solve C-large]
8 points	

Problem

Alex and Bob are brothers and they both enjoy reading very much. They have widely different tastes on books so they keep their own books separately. However, their father thinks it is good to promote exchanges if they can put their books together. Thus he has bought an one-row bookshelf for them today and put all his sons' books on it in random order. He labeled each position of the bookshelf the owner of the corresponding book ('Alex' or 'Bob').

Unfortunately, Alex and Bob went outside and didn't know what their father did. When they were back, they came to realize the problem: they usually arranged their books in their own orders, but the books seem to be in a great mess on the bookshelf now. They have to sort them right now!!

Each book has its own *worth*, which is represented by an integer. Books with odd values of worth belong to Alex and the books with even values of worth belong to Bob. Alex has a habit of sorting his books from the left to the right in an increasing order of worths, while Bob prefers to sort his books from the left to the right in a decreasing order of worths.

At the same time, they do not want to change the positions of the labels, so that after they have finished sorting the books according their rules, each book's owner's name should match with the label in its position.

Here comes the problem. A sequence of **N** values s_0, s_1, ..., s_{N-1} is given, which indicates the worths of the books from the left to the right on the bookshelf currently. Please help the brothers to find out the sequence of worths after sorting such that it satisfies the above description.

Input

The first line of input contains a single integer **T**, the number of test cases. Each test case starts with a line containing an integer **N**, the number of books on the bookshelf. The next line contains **N** integers separated by spaces, representing s_0, s_1, ..., s_{N-1}, which are the worths of the books.

Output

For each test case, output one line containing "Case #**X**: ", followed by t_0, t_1, ..., t_{N-1} in order, and separated by spaces. **X** is the test case number (starting from 1) and t_0, t_1, ..., t_{N-1} forms the resulting sequence of worths of the books from the left to the right.

Limits

1 ≤ **T** ≤ 30.

Small dataset

1 ≤ **N** ≤ 100
-100 ≤ s_i ≤ 100

Large dataset

1 ≤ **N** ≤ 1000
-1000 ≤ s_i ≤ 1000

Sample

```
Input                        Output

2                            Case #1: 1 4 2 3 5
5                            Case #2: -5 88 11 20 2 -12 87
5 2 4 3 1
7
-5 -12 87 2 88 20 11
```

Powered by

Google Cloud Platform

Campus Test
Powered by Google Code Jam

Questions asked

Problem D. **Cross the maze**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 10 points	Solve D-small
Large input 13 points	Solve D-large

Problem

Edison, a robot, does not have a right hand or eyes. As a brave robot, **he always puts his left hand on the wall no matter he walks or turns around**. Because he thinks it is too dangerous, Edison does not walk backward.

Assume that Edison has found himself in a square-shaped maze of **N**x**N** square cells which is surrounded by walls from the outside. In the maze, some of the cells are also walls. Edison can only move between two empty cells in four directions, north, south, west and east. In order to get out of the maze, he drafts a plan. He uses his left hand to lean on the wall and goes by following the wall.

Here is the question, is Edison able to get out of the maze in at most 10,000 steps? If he can make it, output the path. By getting out of the maze, he only needs to be in the exit cell. If the starting cell is the same as the exit, Edison won't need to move and can directly get out of the maze.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with an integer **N**. **N** is the size of the maze. The following **N** lines, each line contains **N** characters which may be '.' or '#'. '.' is an empty cell, '#' is a wall. Followed by a line which contains four integers: **sx**, **sy**, **ex**, **ey**. (**sx**, **sy**) means that Edison is standing on row **sx** and column **sy** as his starting cell, (**ex**, **ey**) is the exit of the maze. (**sx**, **sy**) is guaranteed to be at one of the 4 corners of the maze, and Edison can only touch the wall on 4 adjacent cells(not 8) initially. (**ex**, **ey**) can be anywhere in the maze. Note that the top-left corner is at position (1,1).

Output

For each test case, output a line containing "Case #x: y", where x is the case number (starting from 1) and y is "Edison ran out of energy." (without the quotes) if Edison can't reach the exit of the maze in at most 10,000 steps, otherwise y should be the number of steps followed by another line which contains y characters to describe the path (each character should be E for east, S for south, W for west or N for north). There is no character to represent the turning around. We don't care about the turning around steps, please only output the path of how Edison will cross the maze.

Limits

1 ≤ **T** ≤ 30.
1 ≤ **sx**, **sy**, **ex**, **ey** ≤ **N**.
The starting cell and the exit of the maze will always be an empty cell. And the starting cell and the exit of the maze won't be the same.

Small dataset

2 ≤ **N** ≤ 10.

Large dataset

2 ≤ **N** ≤ 100.

Sample

```
Input       Output

3           Case #1: Edison ran out of energy.
2           Case #2: 22
.#          SEEENSESSSNNNWWSWWSSEE
#.          Case #3: 4
1 1 2 2     EESS
5
.##.#
.....
...#.
.###.
...#.
```

```
1 1 5 3
3
...
.#.
...
1 1 3 3
```

Note:
In the 2nd test case after moving 1 cell down from his starting cell, Edison will
still be able to lean on the wall at the cell (1,2) by his left hand.
In the third test case, due to Edison can't touch the wall at cell (2,2) initially, so
he has to go east in his first step.

Campus Test
Powered by Google Code Jam

☐ Submissions

☐ Top Scores

dreamoon	100
springegg	100
tckwok	100
cgy4ever	100
OR.Director	100
AlanC	100
Mochavic	100
jxwuyi	100
oldherl	100
Descent	100

Problem E. **Spaceship Defence**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the **Quick-Start Guide** to get started.

Small input 10 points	Solve E-small
Large input 14 points	Solve E-large

Problem

The enemy has invaded your spaceship, and only superior tactics will allow you to defend it! To travel around your spaceship, your soldiers will use two devices: *teleporters* and *turbolifts*.

Teleporters allow your soldiers to move instantly between rooms. Every room contains a teleporter, and rooms are color-coded: if a soldier is in a room with some color, she can use the teleporter in that room to immediately move to any other room with the same color.

Turbolifts allow your soldiers to move between rooms more slowly. A turbolift is like an elevator that moves in many directions. Each turbolift moves from one room to one other room, and it takes a certain amount of time to travel. Notes about turbolifts:

- Turbolifts are not two-way: if a turbolift moves soldiers from room a to room b, the same turbolift cannot move soldiers from room b to room a, although there might be another turbolift that does that.
- More than one soldier can use the same turbolift, and they do not interfere with each other in any way.

You will be given the locations and destinations of several soldiers. For each soldier, output the minimum amount of time it could take that soldier to travel from his location to his destination.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

For every test case:

The first line of every test case contains an integer **N**, which is the number of rooms in your spaceship. The rooms are numbered from 1 to **N**. The following **N** lines each contain a string telling the color of the rooms, from room 1 to room **N**. The strings only contain characters a-z (the lower-case English letters) and 0-9 (the number 0 to 9), and the length of each string will be less than or equal to 2.

The next line in the test case is an integer **M**, which indicates the number of turbolifts in your spaceship. The following **M** lines each contain 3 space-separated integers a_i, b_i, t_i, telling us that there is a turbolift that can transport soldiers from room a_i to room b_i in t_i seconds.

The next line in the test case contains an integer **S**, which is the number of soldiers at your command. The following **S** lines each contain two integers: the location and destination of one soldier, p_j and q_j.

Output

For each test case, output one line containing only the string "Case #x:", where x is the number of the test case (starting from 1). On the next **S** lines, output a single integer: on line **j**, the smallest number of seconds it could take for a soldier to travel from p_j to q_j. If there is no path from p_j to q_j, the integer you output should be -1.

Limits

$1 \le$ **S** ≤ 100.
$1 \le a_i, b_i \le$ **N**.
$0 \le t_i \le 1000$.
$1 \le p_j, q_j \le$ **N**.

Small dataset

$1 \le$ **T** ≤ 10.
$1 \le$ **N** ≤ 1000.
$0 \le$ **M** ≤ 3000.

Large dataset

T = 1.
$1 \le$ **N** ≤ 80000.

$0 \le$ **M** ≤ 3000.

Sample

```
Input          Output

3              Case #1:
3              -1
gl             0
t3             Case #2:
t3             -1
3              0
1 2 217        0
3 2 567        Case #3:
1 1 21         3
2              55
2 1            -1
2 3
4
ca
bl
bl
8z
0
3
1 2
2 3
1 1
8
re
b7
ye
gr
0l
0l
ye
b7
7
4 1 19
2 4 21
2 5 317
4 5 34
4 7 3
4 8 265
8 6 71
3
4 3
2 6
1 4
```

Problem A. Sudoku Checker

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 5 points	Solve A-small
Large input 9 points	Solve A-large

Problem

Sudoku is a popular single player game. The objective is to fill a 9x9 matrix with digits so that each column, each row, and all 9 non-overlapping 3x3 sub-matrices contain all of the digits from 1 through 9. Each 9x9 matrix is partially completed at the start of game play and typically has a unique solution.

Given a completed N^2xN^2 Sudoku matrix, your task is to determine whether it is a *valid* solution. A *valid* solution must satisfy the following criteria:

- Each row contains each number from **1** to N^2, once each.
- Each column contains each number from **1** to N^2, once each.
- Divide the N^2xN^2 matrix into N^2 non-overlapping **N**x**N** sub-matrices. Each sub-matrix contains each number from **1** to N^2, once each.

You don't need to worry about the uniqueness of the problem. Just check if the given matrix is a valid solution.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with an integer **N**. The next N^2 lines describe a completed Sudoku solution, with each line contains exactly N^2 integers. All input integers are positive and less than 1000.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is "Yes" (quotes for clarity only) if it is a valid solution, or "No" (quotes for clarity only) if it is invalid. Note that the judge is case-sensitive, so answers of "yes" and "no" will not be accepted.

Limits

1 ≤ **T** ≤ 100.

Small dataset

N = 3.

Large dataset

$3 \le$ **N** ≤ 6.

Sample

```
Input                         Output

3                             Case #1: Yes
3                             Case #2: No
5 3 4 6 7 8 9 1 2             Case #3: No
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 5 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
3
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
3
5 3 4 6 7 8 9 1 2
6 7 2 1 9 5 3 4 8
1 9 8 3 4 2 5 6 7
8 5 9 7 6 1 4 2 3
4 2 6 8 999 3 7 9 1
7 1 3 9 2 4 8 5 6
9 6 1 5 3 7 2 8 4
2 8 7 4 1 9 6 3 5
3 4 5 2 8 6 1 7 9
```

Campus Test
Powered by Google Code Jam

Questions asked

Problem B. Meet and party

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the **Quick-Start Guide** to get started.

| Small input 9 points | Solve B-small |
| Large input 15 points | Solve B-large |

Problem

Little Sin lives in a Manhattan-grid city, a 2D plane where people can only go north, west, south or east along the grid. The distance from (x1, y1) to (x2, y2) is |x1 - x2| + |y1 - y2|.

Little Sin really likes to party and is hoping to host a house party in Manhattan this Sunday. Little Sin has collected a list of people who will attend, and now needs to decide at whose home she will host the party.

Little Sin invited all of the people in several rectangular areas, and all of those people have said yes. A rectangular area is denoted as (x1, y1, x2, y2), where x1 ≤ x2, y1 ≤ y2. People who live in a rectangular area fill all integral points inside it. So there are a total of (x2 - x1 + 1) * (y2 - y1 + 1) people in the rectangular area (x1, y1, x2, y2).

Little Sin knows the coordinates of those rectangular areas. She wants the party to be hosted at the home of one of the people who is attending, but she also doesn't want everyone else to have to travel very far: she wants to minimize the sum of all distances from all attendees' houses to the party. Can you help her?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing a single integer: the number of rectangular areas, **B**. **B** lines follow. Each line contains 4 integers: x1, y1, x2, y2, denoting the coordinates of a rectangular area of people Little Sin has invited to her party.

Output

For each test case, output one line containing "Case #t: x y d", where t is the case number (starting from 1) and (x, y) is the coordinates of the person whose home the party should be hosted. If there are multiple positions with the same minimum total distance, choose the one with the smallest x. If there are still multiple positions, choose the one with the smallest y. The value d is the sum of the distances from all attendees' houses to the point (x, y).

Limits

1 ≤ **T** ≤ 10.
|x1|, |y1|, |x2|, |y2| ≤ 10^9.
x1 ≤ x2, y1 ≤ y2.
The rectangular areas within a test case don't intersect.

Small dataset

1 ≤ **B** ≤ 100.
1 ≤ **Total number of people in each test case** ≤ 1000.

Large dataset

1 ≤ **B** ≤ 1000.
1 ≤ **Total number of people in each test case** ≤ 1000000.

Sample

```
Input            Output

2                Case #1: 1 1 12
1                Case #2: -1 2 6
0 0 2 2
3
-1 2 -1 2
0 0 0 0
1 3 1 3
```

Questions asked

Problem C. **Hex**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

| Small input 12 points | Solve C-small |
| Large input 13 points | Solve C-large |

Problem

This problem was inspired by a board game called Hex, designed independently by Piet Hein and John Nash. It has a similar idea, but does not assume you have played Hex.

This game is played on an **N**x**N** board, where each cell is a hexagon. There are two players: Red side (using red stones) and Blue side (using blue stones). The board starts empty, and the two players take turns placing a stone of their color on a single cell within the overall playing board. Each player can place their stone on any cell not occupied by another stone of any color. There is no requirement that a stone must be placed beside another stone of the same color. The player to start first is determined randomly (with equal probability among the two players).

The upper side and lower sides of the board are marked as red, and the other two sides are marked as blue. The goal of the game is to form a connected path of one player's stones connecting the two sides of the board that have that player's color. The first player to achieve this wins. Note that the four corners are considered connected to both colors.

The game ends immediately when one player wins.

Given a game state, help someone new to the game determine the status of a game board. Say one of the following:

- "**Impossible**": If it was impossible for two players to follow the rules and to have arrived at that game state.
- "**Red wins**": If the player playing the red stones has won.
- "**Blue wins**": If the player playing the blue stones has won.
- "**Nobody wins**": If nobody has yet won the game. Note that a game of Hex can't end without a winner!

Note that in any impossible state, the only correct answer is "Impossible", even if red or blue has formed a connected path of stones linking the opposing sides of the board marked by his or her colors.

Here's a an example game on a 6x6 gameboard where blue won. Blue was the first player to move, and placed a blue stone at cell marked as 1. Then Red placed at cell 2, then blue at cell 3, etc. After the 11th stone is placed, blue wins.
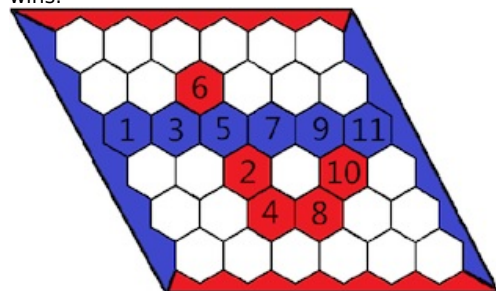
Input

The first line of input gives the number of test cases, **T**. **T** test cases follow. Each test case start with the size of the side of the board, **N**. This is followed by a board of **N** rows and **N** columns consisting of only 'B', 'R' and '.' characters. 'B' indicates a cell occupied by blue stone, 'R' indicates a cell occupied by red stone, and '.' indicates an empty cell.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the status of the game board. It can be "Impossible", "Blue wins", "Red wins" or "Nobody wins" (excluding the quotes). Note that the judge is case-sensitive, so answers of "impossible", "blue wins", "red wins" and "nobody wins" will be judged incorrect.

Limits

$1 \leq$ **T** ≤ 100.

Small dataset

$1 \le N \le 10$.

Large dataset

$1 \le N \le 100$.

Sample

```
Input       Output

7           Case #1: Nobody wins
1           Case #2: Blue wins
.           Case #3: Red wins
1           Case #4: Impossible
B           Case #5: Blue wins
1           Case #6: Impossible
R           Case #7: Blue wins
2
BR
BB
4
BBBB
BBB.
RRR.
RRRR
4
BBBB
BBBB
RRR.
RRRR
6
......
..R...
BBBBBB
..R.R.
..RR..
......
```

Campus Test
Powered by Google Code Jam

Problem D. **Dragon Maze**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 8 points	Solve D-small
Large input 12 points	Solve D-large

Problem

You are the prince of Dragon Kingdom and your kingdom is in danger of running out of power. You must find power to save your kingdom and its people. An old legend states that power comes from a place known as Dragon Maze. Dragon Maze appears randomly out of nowhere without notice and suddenly disappears without warning. You know where Dragon Maze is now, so it is important you retrieve some power before it disappears.

Dragon Maze is a rectangular maze, an **N** x **M** grid of cells. The top left corner cell of the maze is (0,0) and the bottom right corner is (**N**-1, **M**-1). Each cell making up the maze can be either a dangerous place which you never escape after entering, or a safe place that contains a certain amount of power. The power in a safe cell is automatically gathered once you enter that cell, and can only be gathered once. Starting from a cell, you can walk up/down/left/right to adjacent cells with a single step.

Now you know where the entrance and exit cells are, that they are different, and that they are both safe cells. In order to get out of Dragon Maze before it disappears, you must walk from the entrance cell to the exit cell taking as few steps as possible. If there are multiple choices for the path you could take, you must choose the one on which you collect as much power as possible in order to save your kingdom.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow.

Each test case starts with a line containing two integers **N** and **M**, which give the size of Dragon Maze as described above. The second line of each test case contains four integers en_x, en_y, ex_x, ex_y, describing the position of entrance cell (en_x, en_y) and exit cell (ex_x, ex_y). Then **N** lines follow and each line has **M** numbers, separated by spaces, describing the **N x M** cells of Dragon Maze from top to bottom. Each number for a cell is either -1, which indicates a cell is dangerous, or a positive integer, which indicates a safe cell containing a certain amount of power.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1). If it's possible for you to walk from the entrance to the exit, y should be the maximum total amount of power you can collect by taking the fewest steps possible. If you cannot walk from the entrance to the exit, y should be the string "Mission Impossible." (quotes for clarity). Please note that the judge requires an exact match, so any other output like "mission impossible." or "Mission Impossible" (which is missing the trailing period) will be judged incorrect.

Limits

The amount of power contained in each cell will not exceed 10,000.
$1 \le$ **T** ≤ 30.
$0 \le$ **en_x, ex_x** $<$ **N**.
$0 \le$ **en_y, ex_y** $<$ **M**.

Small dataset

$1 \le$ **N**, **M** ≤ 10.

Large dataset

$1 \le$ **N**, **M** ≤ 100.

Sample

Input	Output
2	Case #1: Mission Impossible.
2 3	Case #2: 7
0 2 1 0	
2 -1 5	

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

Powered by

Google Cloud Platform

Campus Test
Powered by Google Code Jam

Questions asked

Submissions

Sudoku Checker

| 5pt | Not attempted **1471/2010 users** correct (73%) |
| 9pt | Not attempted **1146/1443 users** correct (79%) |

Meet and party

| 9pt | Not attempted **496/823 users** correct (60%) |
| 15pt | Not attempted **47/409 users** correct (11%) |

Hex

| 12pt | Not attempted **19/260 users** correct (7%) |
| 13pt | Not attempted **14/18 users** correct (78%) |

Dragon Maze

| 8pt | Not attempted **336/594 users** correct (57%) |
| 12pt | Not attempted **229/330 users** correct (69%) |

Ignore all my comments

| 17pt | Not attempted **216/468 users** correct (46%) |
| 0pt | Not attempted |

Top Scores

TankEngineer	100
Nekosyndrome	100
l521530	100
W.Junqiao	100
LTzycLT	100
iloahz	100
drazil	87
navi	85
wishstudio	85
redsniper	76

Problem E. Ignore all my comments

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

| Small input 17 points | Solve E-small |
| Large input 0 points | Solve E-large |

Problem

Good programmers write fabulous comments. Igor is a programmer and he likes the old C-style comments in **/* ... */** blocks. For him, it would be ideal if he could use this style as a uniform comment format for all programming languages or even documents, for example Python, Haskell or HTML/XML documents.

Making this happen doesn't seem too difficult to Igor. What he will need is a comment pre-processor that removes all the comment blocks in **/***, followed by comment text, and by another ***/**. Then the processed text can be handed over to the compiler/document renderer to which it belongs—whatever it is.

Igor's pre-processor isn't quite that simple, though. Here are some cool things it does:

• The comments the pre-processor reads can be nested the same way brackets are nested in most programming languages. It's possible to have comments inside comments. For example, the following code block has an outer level of comments that should be removed by the comment pre-processor. The block contains two inner comments.

```
printf("Hello /* a comment /* a comment inside comment */
        inside /* another comment inside comment */
        string */ world");
```

After the pre-process step, it becomes:

```
printf("Hello  world");
```

• Igor recognizes comments can appear anywhere in the text, including inside a string "/*...*/", a constant number 12/*...*/34 or even in a character escape \/*...*/n

Or more formally:

```
text:
   text-piece
   text-piece remaining-text
text-piece:
   char-sequence-without-/*
   empty-string
remaining-text:
   comment-block text

comment-block:
   /* comment-content */
comment-content:
   comment-piece
   comment-piece remaining-comment
comment-piece:
   char-sequence-without-/*-or-*/
   empty-string
remaining-comment:
   comment-block comment-content

char:
   letters
   digits
   punctuations
   whitespaces
```

Our pre-processor, given a **text**, removes all **comment-block** instances as specified.

Notes

• Igor only needs to remove the comment in one pass. He doesn't remove additional comment blocks created as a result of the removal of any comment

block. For example:

```
//*no recursion*/* file header */
```

should generate:

```
/* file header */
```

- The ***** character in any **/*** or **/***cannot be re-used in another **/*** or ***/**. For example the following does **NOT** form a proper comment block

```
/*/
```

Input

A text document with comment blocks in **/*** and ***/**. The input file is valid. It follows the specification of **text** in the problem statement. The input file always terminates with a newline symbol.

Output

We only have one test case for this problem. First we need to output the following line.

```
Case #1:
```

Then, print the document with all comments removed, in the way specified in the problem statements. Don't remove any spaces or empty lines outside comments.

Limits

The input program contains only:

- Letters: a-z, A-Z,
- Digits: 0-9
- Punctuation: ~ ! @ # % ^ & * ( ) - + = : ; " ' < > , . ? | / \ { } [ ] _
- Whitespace characters: space, newline

Small dataset

The small input contains a program of less than 2k bytes.

Large dataset

The large input contains a program of less than 100k bytes.

Sample

Input

```
//*no recursion*/* file header
***********/************
* Sample input program *
**********/*************
*/
int spawn_workers(int worker_count) {
  /* The block below is supposed to spawn 100 workers.
     But it creates many more.
     Commented until I figure out why.
  for (int i = 0; i < worker_count; ++i) {
    if(!fork()) {
      /* This is the worker. Start working. */
      do_work();
    }
  }
  */
  return 0; /* successfully spawned 100 workers */
}

int main() {
  printf("Hello /*a comment inside string*/ world");
  int worker_count = 0/*octal number*/144;
  if (spawn_workers(worker_count) != 0) {
    exit(-1);
  }

  return 0;
}
```

Output

```
Case #1:
/* file header
************************
*/
int spawn_workers(int worker_count) {

  return 0;
}

int main() {
  printf("Hello  world");
  int worker_count = 0144;
  if (spawn_workers(worker_count) != 0) {
    exit(-1);
  }
  return 0;
}
```