The 36th ACM-ICPC Asia Shanghai Regional Contest

hosted by Fudan University

Problem Set

Online Version for UVa OJ

A. Zombie's Treasure Chest

[Description]

Some brave warriors come to a lost village. They are very lucky and find a lot of treasures and a big treasure chest, but with angry zombies.

The warriors are so brave that they decide to defeat the zombies and then bring all the treasures back. A brutal long-drawn-out battle lasts from morning to night and the warriors find the zombies are undead and invincible.

Of course, the treasures should not be left here. Unfortunately, the warriors cannot carry all the treasures by the treasure chest due to the limitation of the capacity of the chest. Indeed, there are only two types of treasures: emerald and sapphire. All of the emeralds are equal in size and value, and with infinite quantities. So are sapphires.

Being the priest of the warriors with the magic artifact: computer, and given the size of the chest, the value and size of each types of gem, you should compute the maximum value of treasures our warriors could bring back.

[Input]

There are multiple test cases. The number of test cases T (T <= 200) is given in the first line of the input file. For each test case, there is only one line containing five integers N, S1, V1, S2, V2, denoting the size of the treasure chest is N and the size and value of an emerald is S1 and V1, size and value of a sapphire is S2, V2. All integers are positive and fit in 32-bit signed integers.

[Output]

For each test case, output a single line containing the case number and the maximum total value of all items that the warriors can carry with the chest.

[Sample Input]

```
2
100 1 1 2 2
100 34 34 5 3
```

[Sample Output]

```
Case #1: 100
Case #2: 86
```

B. Yummy Triangular Pizza

[Description]

Pizzahat has released a new pizza with triangular shaped pieces. This pizza is composed of some equal-sized equilateral triangle. Moreover, all the triangles are connected. Also, if two triangles are directly connected, they must share a common edge.

How many different shapes of this kind of N-pieces pizza are there? Two patterns are considered as same if they can completely overlap after rotation and shifting (note that flipping is not included).

[Input]

There are multiple test cases. The first line of input contains a single integer denoting the number of test cases.

For each test case, there is only one line with only one integer N denoting the number of pieces that can be used. (1 <= N <= 16)

[Output]

For each test case, output a single integer denoting the number of possible different shapes of the pizza.

[Sample Input]

```
3
2
4
10
```

[Sample Output]

```
Case #1: 1
Case #2: 4
Case #3: 866
```

C. Xavier is Learning to Count

[Description]

Xavier, a 9-year-old student, loves playing many kinds of puzzles. One of his favourites is the following:

Xerier, his classmate, has made many cards. She writes down a single positive number on each of them. No numbers written on different cards are the same. After that she writes down an equation, whose right side is a single positive number chosen by her, and the left side is the sum of p integers:

$$X_1 + X_2 + \cdots + X_p = n$$

Then she asks Xavier put p cards on the corresponding X_i's position to make this equation correct, with an additional condition that X_i should be ordered from smaller to bigger, i.e.

$$X_i < X_{i+1}, \forall 1 \leq i < p$$

Every time Xavier immediately comes up with many solutions. Now he wants to know how many solutions in total are there for any n given by Xerier.

[Input]

There are multiple test cases. The number of them is given in the beginning of the input. Then a series of input block comes one by one.

For each test case:

The first line contains two space-separated integers m and p (1<=p<=5). The second line contains m distinct positive integers - the numbers written on each of the cards. None of these integers exceeds 13000.

There are about 120 test cases in total, but 90% of them are relatively small. More precisely, all numbers are less than or equal to 100 in 90% of the test cases.

[Output]

For each test case:

For each positive integer, output the number of ways in a single line. To keep the output finite, only numbers with positive ways should be outputted.

Output a blank line after each test case. See sample for more format details.

[Sample Input]

```
3
3 3
```

```
1 2 3
5 4
1 3 5 6 7
10 3
1 2 3 4 5 6 7 8 9 10
```

[Sample Output]

```
Case #1:
6: 1

Case #2:
15: 1
16: 1
17: 1
19: 1
21: 1

Case #3:
6: 1
7: 1
8: 2
9: 3
10: 4
11: 5
12: 7
13: 8
14: 9
15: 10
16: 10
17: 10
18: 10
19: 9
20: 8
21: 7
22: 5
23: 4
24: 3
25: 2
26: 1
27: 1
```

D.Winmine.exe

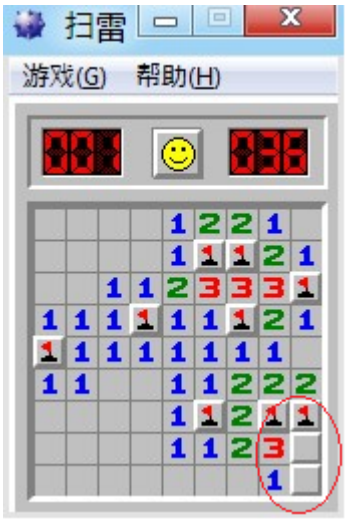
[Description]

The Windows Minesweeper (WinMine) is one of the most well-known games on the Windows system. The rule is quite easy and it takes only a few minutes to play a set. First of all, let's briefly introduce this game (if you believe that you are familiar enough with this game, you can skip the next paragraph):

The goal of the game is to uncover all the squares that do not contain mines (with the left mouse button) without being "destroyed" by clicking on a mine. The location of the mines is discovered by a process of logic. Clicking on the game board will reveal what is hidden underneath the chosen square or squares (a large number of blank squares may be revealed in one go if they are adjacent to each other). Some squares are blank but some contain numbers (1 to 8), each number being the number of mines adjacent to the uncovered square. The game is won once all blank squares have been uncovered without hitting a mine, any remaining mines not identified by flags being automatically flagged by the computer. The distinctive feature of minesweeper is the randomness at the initial stage and at some intermediate stages.

----Wikipedia

Jaddy loves playing WinMine at free time very much due to its simplicity and ingenuity. However, sometimes it makes him frustrated when he reaches some undeterminable states during the game. See the following state as an example:



In the red circle of this example, the rest two squares are absolutely undeterminable and there are apparently two possible distributions of the rest ONE mine.

When Jaddy gets into this kind of trouble, he has no choice but to guess. Sometimes there are only two possible choices (as what the example says) but sometimes there are many. This way, he needs an assistant of this game to calculate the number of different possible distributions of mines depending on the current state of game board.

This assistant of game is pretty useful and interesting, at least for him, so that he immediately starts coding. Unfortunately, Jaddy spent all the time to play WinMine in the class of Programming and Algorithm, so he feels such a complex task is a mission impossible. Jaddy loves WinMine very much so that he asks you, an excellent programmer, for help. In addition, in order to reduce the difficulty, **he added three constraints**:

1. It is guaranteed that the input state of game can always be created by making only **ONE** click on the initial board.
2. In the given state, if two unrevealed squares are connected directly or by any other unrevealed squares, they are also bi-connected. That is to say, even after any one of the other unrevealed squares is removed, these two squares are still connected. Here we say two squares are connected if and only if they **share an edge**. That means a square has at most four squares connected.
3. In the given state, if two numbers are connected by sharing an edge directly or by other numbers, all the unrevealed cells adjacent to the connected set of numbers must be connected.

Help Jaddy please!

[Input]

There are multiple test cases, the first line of input is a positive integer T (T <= 50) indicating the number of test cases. Then T cases follow. For each test case, the first line contains three positive integers n, m and M where 1 <= n, m <= 100 and 1 <= M <= 1500 denoting the number of rows and columns in the given board and the total number of mines on the board. Then an n*m matrix of chars describing the current state of board will be given. In this matrix, there are 2 styles of symbols:

1. digits from 0 to 8: that means the number of mines around this square (‘0’ implies a blank square).
2. ‘.’ (quotes for clarifying): that means this square is unrevealed.

[Output]

For each test case, output an integer leading by the case number, denoting the number of possible distribution of mine in the given state, module by

1000003. See the sample output for further details.

[Sample Input]

```
2
4 5 2
…..
.....
11111
00000
4 5 4
…..
.....
22222
00000
```

[Sample Output]

```
Case #1: 2
Case #2: 1
```

E. Very Boring Homework

[Description]

Prof. Z thinks his homework is very hard for most of his students to solve (do you remember the task “Boring Homework”?) To his surprise, many students hand in correct solutions. He thinks the reason is actually the small size of the data set he used to test students’ programs rather than the low difficulty of the homework task. He decides to give his students the same homework again, but with enormous test cases. Of course, his students think his homework becomes even more boring this time. They need your help again.

For the ones who don’t know what homework Prof. Z. had given to his students last time:

You’re asked to draw a graph of a binary search tree (BST).

A binary search tree, which may sometimes also be called ordered or sorted binary tree, is a node-based binary tree data structure which has the following properties:

The left subtree of a node contains only nodes with keys less than the node's key.

The right subtree of a node contains only nodes with keys greater than the node's key.

Both the left and right subtrees must also be binary search trees.

--from Wikipedia

Given a list of integer keys that should be inserted into the BST one by one orderly, we can form a unique BST. Moreover, Prof. Z wants the students to draw the graph of this BST.

The rules to draw a graph of a BST are listed below:

1. The graph of a 1-node BST is a single 'o' (15th small Latin letter).
2. If a BST has a non-empty subtree, draw a single '|' just above the subtree's root, and a single '+' just above the previous drawn '|'. Finally, in the row of '+', use the least number (including 0) of '-'s to connect '+' (corresponding to the left or right subtree) and 'o' (denoting the parent node of the subtrees).
3. The left subtree (if exists) must be drawn on the left side of its parent. Similarly, the right subtree (if exists) must be drawn on the right side of its parent.
4. The column of the BST's root must not contain any characters belonging to left or right subtree.

5. For each node of the BST, the graph of its left subtree and the graph of its right subtree do not share common columns in the picture of the whole tree.

After the whole BST has been drawn, we number the rows from top to bottom, counting from 1, and so do the columns from left to right, counting from 1.

Due to the large scale of the tree, the graph will become so large that it is impossible for Prof. Z to check every detail of the graph this time. So you are only asked to hand in m fragments of that graph to Prof. Z instead of the whole one.

[Input]

The first line contains T, the number of test cases. T test cases follow.

For each test case:

The first line contains a positive integer N (N <= 100000).

The second line contains N distinct integers, each of which can be represented by a 32-bit signed integer. These numbers should be inserted into an empty BST one by one in the given order.

The third line contains an integer M (M <= 5).

M lines follow, each contains four integers, which are the row and column number of the top left corner, and the number of rows R_i and columns C_i of the required graph fragment, respectively. All the input integers will be positive and fit into a 32-bit signed integer, except R_i and C_i, which will be less than or equal to 200 and greater than 0.

[Output]

For each test case:

Output the case number counting from 1 in the first line.

Then M blocks follow, each contains R_i (or less, see next) lines. Each line should contain exactly C_i characters. Use space (ASCII 32) to fill in the blank. But if a line contains only whitespaces, this line should not be outputted.

Output a blank line after each graph fragment.

[Sample Input]

```
3
3
3 1 2
1
1 1 5 3
```

```
6 4 5 6 1 3 2
1
1 1 8 10
10
2 6 7 4 5 3 1 9 10 8
2
1 1 5 5
3 6 5 5
```

[Sample Output]

```
Case #1:
+-o
|
o+
 |
 o

Case #2:
+--o+
|   |
o-+ o+
  |  |
 +o  o
 |
 o

Case #3:
+o---
|
o  +-
   |
  +o+

o+
 |
 o-+
   |
  +o+
```

F. Universal Question Answering System

[Description]

Every student needs help from getting new knowledge by asking questions. Surveys are suggesting that some similar questions are repeated frequently. So it will be nice to develop an automatic question-answering system to answer these questions. Your algorithm should not have any prior knowledge, but it must be able to read sentences and remember the mentioned facts. Whenever the question is asked about such a fact, the system has to answer it properly.

[Input]

The input consists of many dialogues.

There is a single positive integer T on the first line of input. (T <= 500, but note that 95% of them are relatively small) It denotes the number of following dialogues. Each dialogue includes one or more lines. Each line contains one sentence: either a statement or a question. Statements end with a dot character (.) while questions end with a question mark (?). There is one extra line after each dialogue. That line ends with an exclamation mark (!). The definitions of the statements and questions will be discussed later.

Sentences can contain words, spaces and punctuation characters. All words contain only Latin letters and are case-sensitive. Unlike the normal English writing rules, the first letter of a sentence should keep lowercase unless the first word itself should begin with a capital letter. There are no extra spaces between words. No word will have more than 10 characters. There will be at most 1000 lines per dialogue.

Statements

Each statement has one of the following forms:

noun_phrase are noun_phrase.
noun_phrase can verb_phrase.
everything which can verb_phrase can verb_phrase.
everything which can verb_phrase are noun_phrase.

noun_phrase and verb_phrase are both single word. The meanings of the four forms are:

A are B: If X is A, then X is B.

A can B: If X is A, then X has the ability to B.
everything which can A can B: If X has the ability to A, X has the ability to B.
everything which can A are B: If X has the ability to A, X is B.

Questions
Each question has one of the following forms:

are noun_phrase noun_phrase?
can noun_phrase verb_phrase?
can everything which can verb_phrase verb_phrase?
are everything which can verb_phrase noun_phrase?

They are the question form of the statements.

In each test case, the number of different noun phrases will not exceed 100; the number of different verb phrases will not exceed 100.

[Output]

For each test case, output two lines. The first line describes the test case number counting from 1, while the second line contains the same number of characters as the number of questions in this test case. Each character is either 'Y'(denoting you can get that fact logically) or 'M'(otherwise), without quotes.

[Sample Input]

```
1
flies can fly.
flies are insects.
everything which can fly are animals.
are everything which can fly insects?
are flies animals?
can flies eat?
everything which can fly can eat.
can flies eat?
Bye!
```

[Sample Output]

```
Case #1:
MYMY
```

G. Triangles and Quadrangle

[Description]

Little Yuan is two years old and she is learning about some triangles and quadrangles. She is such a smart girl that she soon realizes two triangles can form a quadrangle without overlapping each other. She picks up a lot of triangles and uses them to form some quadrangles. Unfortunately, she is not good at this kind of jigsaw game and makes some mistakes.

As her brother, you are curious about whether she has made a mistake when forming two triangles into a quadrangle. You are thinking about to write a program to determine it.

Notice that the quadrangle in this problem is defined as a simple polygon with four vertexes. And you may also assume that all triangles and quadrangle have positive area.

Note that two graphs are considered as the same if and only if they can overlap completely by shifting, rotation and flipping. You can also shifting, rotation and flipping one of the triangles to form the triangles to a quadrangle.

[Input]

There are multiple test cases. The first line of input contains a single integer T denoting the number of test cases. (T <= 500)

For each test case, there are 10 lines in total.

The first 3*2 lines describe the two triangles. Each line with two integers denotes the coordinates of a point.

The next 4 lines describe the quadrangle in clockwise order or counter-clockwise order.

All coordinates are less than 15000 in absolute value.

[Output]

For each test case, output “Yes” if the given triangles can form a quadrangle without overlapping, “No” otherwise.

[Sample Input]

```
2
0 0
0 1
1 0
```

```
0 0
0 1
1 0
-1 0
0 0
1 0
0 1
0 0
-1 1
1 0
0 0
0 1
1 0
-1 0
0 0
1 0
0 1
```

[Sample Output]

```
Case #1: Yes
Case #2: No
```

H.Share the Cakes

[Description]

Lunar Rose is a pure and kind girl who was born in a serene town on, with surprising coincidence, the Mid-autumn Festival that year. Therefore, on each of her birthday, she enjoys two cakes: for the birthday, and for the festival.

This year, Lunar would like to share the cakes with Jaddy, her boyfriend, on the nice day, isn't it so romantic? What is the tragedy, Jaddy screwed up, again:

When Lunar placed two cakes on the table, she wanted Jaddy to cut the cakes for her so that they can both have half of the moon cake and so do the birthday cake. But Jaddy, as a lazy and impetuous guy, he just used a knife to cut them together easily, hence although the two cakes had been cut into two pieces each, they were not really uniform. What was worth? Lunar is a pretty perfectionist and suddenly quite angry with Jaddy due to his arbitrariness. Finally, Lunar chose to leave him. Stupid Jaddy!

Jaddy became extremely regretful and distressing. Lunar, a softhearted girl, does not have the heart to make him so painful so that she'd like to give Jaddy a chance to do it again. However, she set a strict limit for this task: Jaddy can only use one cut to separate the two cakes into equivalent halves. Evil Ms. Rose!

Both two cakes are convex polygons; the table and knife are both big enough and long enough thus can be considered as infinite planar and line. This way, tell Jaddy a strategy, which means a line, satisfying Lunar's condition to cut cakes so that he can get her back.

[Input]

The first line of the input data is a positive integer T (T <= 100) indicating the number of test cases. Then T cases follow. Each test case contains description of two cakes (polygons): each polygon begins with an integer n which denotes the number of vertices, and then n pairs of integers (x, y) follows describing the coordinates of vertices counterclockwise. You may assume that each polygon has no more than 20 vertices and all the coordinates are in range of [-1000, 1000]. Besides, the two polygons can be separated by a line, without touching on any points of the line.

[Output]

For each test case, output two real numbers k and b, leading by the case number, which means that Jaddy should cut the cakes along the line y=k*x + b. It is guaranteed that both k and b among the answers are in range of [-10000,

10000]. See sample output for further details. Any answer with 10^-4 relative or absolute error is acceptable.

[Sample Input]

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

[Sample Output]

```
Case #1: 0 1
Case #2: 1 0
```

I. Revenge of Fibonacci

[Description]

The well-known Fibonacci sequence is defined as following:

$$F(0) = F(1) = 1$$
$$F(n) = F(n-1) + F(n-2)\ \forall n \geq 2$$

Here we regard n as the index of the Fibonacci number F(n).

This sequence has been studied since the publication of Fibonacci's book *Liber Abaci*. So far, many properties of this sequence have been introduced.

You had been interested in this sequence, while after reading lots of papers about it. You think there's no need to research in it anymore because of the lack of its unrevealed properties. Yesterday, you decided to study some other sequences like Lucas sequence instead.

Fibonacci came into your dream last night. "Stupid human beings. Lots of important properties of Fibonacci sequence have not been studied by anyone, for example, from the Fibonacci number 347746739…"

You woke up and couldn't remember the whole number except the first few digits Fibonacci told you. You decided to write a program to find this number out in order to continue your research on Fibonacci sequence.

[Input]

There are multiple test cases. The first line of input contains a single integer T denoting the number of test cases (T<=50000).

For each test case, there is a single line containing one non-empty string made up of at most 40 digits. And there won't be any unnecessary leading zeroes.

[Output]

For each test case, output the smallest index of the smallest Fibonacci number whose decimal notation begins with the given digits. If no Fibonacci number with index smaller than 100000 satisfy that condition, output -1 instead – you think what Fibonacci wants to told you beyonds your ability.

[Sample Input]

```
15
1
12
123
```

```
1234
12345
9
98
987
9876
98765
89
32
51075176167176176176
347746739
5610
```

[Sample Output]

```
Case #1: 0
Case #2: 25
Case #3: 226
Case #4: 1628
Case #5: 49516
Case #6: 15
Case #7: 15
Case #8: 15
Case #9: 43764
Case #10: 49750
Case #11: 10
Case #12: 51
Case #13: -1
Case #14: 1233
Case #15: 22374
```

J. Quelling Blade

[Description]

Mr. Sheep lost himself in a computer game. In this game, he plays the part of a super hero and fight with the evil. The equipment is very important in this game and Mr. Sheep thinks the Quelling Blade is the most powerful weapon.

In this game, each type of weapon costs hero some money, and brings the hero benefits. If the hero buys two weapons (no matter they have the same type or not), the benefit values are accumulated. That is to say, if the hero buys two weapons with benefit 3 and 5, the hero will get total benefit value 8=3+5.

There are some requirements for each weapon. If the hero wants to buy a certain weapon, he may need some other weapons first. For example, if hero wants to buy a "Divine Rapier", he needs a "Demon Edge" and a "Scared Relic". Of course, if he wants to buy the second "Devine Rapier", he must buy another "Demon Edge" and another "Scared Relic" first. Notice that the existing weapon will not disappear after the trade. Note that a weapon may need multiple weapons with same type. And you may assume that a type of weapon is required by at most one other type of weapon.

The hero is busy with combat and has no time to earn money. Fortunately, the game will give the hero 1 coin per second. So if the hero wants to buy a "Quelling Blade", the minimum total time for him to achieve his goal can be easily calculated.

Mr. Sheep is a perfectionist. He not only wants to get the "Quelling Blade" as soon as possible, but also wants to optimize every second during the game. He defines the utility of the game as the sum of the benefit value of the hero in each second. He calculates the utility from the start of the game until the second he gets "Quelling Blade", exclusively. You may refer to the samples for further clarification. In the other words, you should define a way of process to buy the weapons for the hero, which minimize the total time to get "Quelling Blade" and optimize the utility of the game.

[Input]

There are hundreds of test cases, the number of test case are in the first line of the input. Notice that most of the test cases are relatively small.

For each test case, the first line contains a single integer N denoting the number of different types of weapons. (1 <= N <= 1000)

The next lines are describing the weapons. For each weapon, the first line contains two integers B and C, denoting the benefit value and the cost of this kind

of weapon. (1 <= B, C <= 2^31-1) Then a single integer P in the next line describes the number of requirements of this weapon. Next P lines, each line contains two integers I and A, means that this weapon needs A weapons of index I.

The indexes of weapons are start from 1 to N. The “Quelling Blade” is the first type of weapon. And you may assume that the total numbers of weapons which are needed by the “Quelling Blade” is less than 1000000. Also notice that “Quelling Blade” can be brought in a finite game time and a type of weapon can be required by at most one other type of weapon.

[Output]

For each test case, output a single integer denoting the optimal utility. You may assume that the answer is fit in 64-bit signed integer.

[Sample Input]

```
2
3
1 1
1
2 2
2 1
1
3 1
1 1
0
3
1 1
1
2 2
1 1
1
3 1
2 1
0
```

[Sample Output]

```
Case #1: 14
Case #2: 17
```