1001

A Simple Problem with Integers

Description:

Let A_1, A_2, ... , A_N be N elements. You need to deal with two kinds of operations. One type of operation is to add a given number to a few numbers in a given interval. The other is to query the value of some element.

Input:

There are a lot of test cases.

The first line contains an integer *N*. (1 <= *N* <= 50000)

The second line contains *N* numbers which are the initial values of A_1, A_2, ... , A_N.

(-10,000,000 <= the initial value of A_i <= 10,000,000)

The third line contains an integer Q. (1 <= Q <= 50000)

Each of the following Q lines represents an operation.

"1 a b k c" means adding c to each of A_i which satisfies a <= i <= b and (i - a) % k == 0. (1 <= a <= b <= N, 1 <= k <= 10, -1,000 <= c <= 1,000)

"2 a" means querying the value of A_a. (1 <= a <= N)

Output:

For each test case, output several lines to answer all query operations.

Sample Input:

```
4
1 1 1 1
14
2 1
2 2
2 3
2 4
1 2 3 1 2
2 1
2 2
2 3
2 4
1 1 4 2 1
2 1
2 2
2 3
2 4
```

Sample Output:

```
1
1
```

1
1
1
3
3
1
2
3
4
1

1002

Alice and Bob

Description:

Alice and Bob's game never ends. Today, they introduce a new game. In this game, both of them have N different rectangular cards respectively. Alice wants to use his cards to cover Bob's. The card A can cover the card B if the height of A is not smaller than B and the width of A is not smaller than B. As the best programmer, you are asked to compute the maximal number of Bob's cards that Alice can cover.

Please pay attention that each card can be used only once and the cards cannot be rotated.

Input:

The first line of the input is a number T (T <= 40) which means the number of test cases.

For each case, the first line is a number N which means the number of cards that Alice and Bob have respectively. Each of the following N (N <= 100,000) lines contains two integers h (h <= 1,000,000,000) and w (w <= 1,000,000,000) which means the height and width of Alice's card, then the following N lines means that of Bob's.

Output:

For each test case, output an answer using one line which contains just one number.

Sample Input:

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

Sample Output:

```
1
2
```

1003

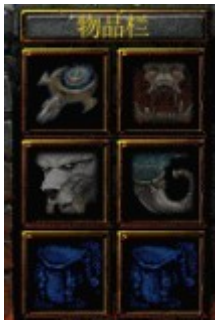
Defend Jian Ge

Description:

Defend Jian Ge, an interesting RPG map in WAR3, has a very complete equipment system as a mini RPG game.

In the game, each player has a backpack which has 6 grids and some gold reserves. There are three kinds of equipments in the game.

1. Normal equipment: Each equipment occupies one grid in the backpack and the player can buy it directly. The value of the normal equipment is equal to its price.
2. Mixture equipment: Each equipment occupies one grid in the backpack and the player can only get it through the synthesis of the corresponding recipe. If you have enough equipment of recipes in your backpack and gold to pay synthesis cost, you can get it. The value of the mixture equipment is equal to the sum of all equipments' value in the recipe plus the synthesis cost. A mixture equipment can be made from several normal equipments and mixture equipments.
3. Consume equipment: A kind of equipment must occupy one grid and the player can buy it directly. That is to say, if you have a number of consume equipments, they must be in one grid. The value of each consume equipment is equal to its price, and when you want to sell this kind of equipment, you must sell the whole grid's equipments at the same time.



Backpack is not full

Backpack is full

Consume equipment

You should pay attention:

1. When the backpack is full, you cannot buy anything.
2. When a mixture equipment is get, the mixture equipment of recipe in you backpack will disappear.
3. If one operation is illegal, there is nothing happened in your backpack and gold.
4. Initially, there is nothing in your backpack and you have 0 gold.
5. The gold you have cannot be a negative number.

As a DS programmer, you want to simulate the dynamic states in your backpack by program. Now you have initial state and several operations, you wonder the final state.

Input:

There are multiple test cases.
The first line contains an integer N_1 (0 <= N_1 <= 20) indicating the kind of normal equipment.
The next N_1 lines each line contains a string and an integer indicating the name of this normal equipment and its price respectively.
**Format: str num
The following line contains an integer N_2 (0 <= N_2 <= 20) indicating the kind of mixture equipment.
Each of the next N_2 lines begins with a string and an integer indicating the name of this mixture equipment and its synthesis cost. Following, the synthesis recipe of this kind of equipment: some pairs of string and integer indicating which kind of equipment and the number you need to synthesis.
**Format: str num: str_1 num_1, str_2 num_2, ... , str_n num_n (num_1 + num_2 + … +num_n<=6 and num_i >= 0 (1 <= i <= n))
The next line contains an integer N_3 (0 <= N_3 <= 20) indicating the kind of consume equipment.
Each of the next N_3 lines contains a string and an integer indicating the name of this consume equipment and it's price.
**Format: str num
The next line contains an integer M (0 <= M <= 100) indicating the number of operation.
Each of the next M lines contains an operation.
There are three kinds of operation:
1. +num indicating you get num gold (0 <= num <= 1000).
2. +str indicating you want to get an equipment whose name is str.
3. -str indicating you want to sell the equipment whose name is str. If you sell the equipment, you can get gold that is the same to its value.

There is a blank line after each case.
All strings of name only contain lowercase character and its length is no more than 15. The price of the equipment is a non-negative integer which is no more than 1000.

Output:
For each case you should output several lines.
The first line output "Case " + case number + ":".
The next line output an integer indicating the number of gold at last.
The next line output an integer k indicating how many grids occupied.
Each of the next k lines contain a string and an integer indicating the name and the number of the equipment. You should output according to the lexicographic.
**Format: str: num
You can get more details from the sample below.
Note: Output a blank line after each test case.

Sample Input:
2
ring 1

sword 2
1
knife 3: ring 1, sword 1
1
medicine 1
4
+100
+ring
+sword
+knife

1
shoe 0
1
wing 1:
1
medicine 1
3
+10
+shoe
+wing

1
shoe 0
1
wing 1:
1
medicine 1
4
+10
+shoe
+wing
-wing

1
shoe 1
1
wing 1: shoe 1
1
medicine 1
8
+100
+shoe
+shoe

+shoe
+shoe
+shoe
+medicine
+medicine

Sample Output:

Case 1:
94
1
knife: 1

Case 2:
9
2
shoe: 1
wing: 1

Case 3:
10
1
shoe: 1

Case 4:
94
6
medicine: 1
shoe: 1
shoe: 1
shoe: 1
shoe: 1
shoe: 1

1004

Dynamic Lover

Description:

As a famous person with universal love, changing girlfriend too often makes me harassment because I can't keep their name in mind timely. To remember who is my lover now, I buy a magic password-box from a wizard.

As a faithful atheist, I do not believe that it is caused by magic power. By doing a deep research, I find that all "magic" factors are just because a small software inside the box. Because the software only uses some simple data structure, it has a dissatisfied complexity. Now I shortly introduce the principle of the software.

We will support a string with dynamic length and three kinds of operations on it. You can assume that we will always have an initial string.

*1. You will receive a short string, and you should connect it after the original string to make the new string.

*2. You will receive an integer len, and you should answer the query: for each index i (1 <= i <= LEN(nowString)), we will get a sub-string from i to i + len - 1 (if i + len - 1 > LEN(nowString), you should make the suffix from index i as its sub-string). You should output the index i whose sub-string has minimum lexicographic.

*3. You will receive an integer len. You should delete the suffix whose length is len from the string now and get a new string.

As a former ACMer, I want to make a new production with better execution speed. But I am not good at data structure, so I need your help.

Input:

There are multiple test cases.

The first line contains a string as the initial string.

The next line contains an integer m (1 <= m <= 100,000) indicating the number of the operations.

Each of the next m lines begins with an integer k (1 <= k <= 3) indicating the kind of operation. If k = 1, it is followed by a short non-empty string, otherwise it is followed by an integer len.

We guarantee that the total length of the initial string and all short strings are not more than 100,000. And for each query, the len is not more than 1000 and the total sum of len is not more than 100,000.

All characters in the input are lower case.

Output:

For each test case, you should output several lines.

For each query, you should output a line indicating the index i whose sub-string has minimum lexicographic. If more answers exist, output the minimal index.

Sample Input:

aacbab
5
2 2
1 aaa
2 3
3 2
2 3

Sample Output:

1
9
7

1005

Find Black Hand

Description:

I like playing game with my friend, although sometimes looks pretty naive. Today I invent a new game called find black hand. The game is not about catching bad people but playing on a string.

Now I generate a string S and several short ones s[i], and I define three kinds of operation.

1. Delete: remove the ith character.
2. Insert: in any position, insert a character as you like.
3. Change: change the ith character into another character as you like.

For each short string s[i], we define a function f(i): after several operation on S, we can find a substring of S same to s[i], and f(i) is the minimal number of operation to achieve it. It looks so native that every one of you can solve f(i) perfectly I think. So I join the string S from end to end, and f(i) changes nothing. So the string "bb" is also a substring of string "baaab".

The "black hand" is the short string s[i] whose f(i) is minimal. Now it's your time to find the black hand.

Input:

There are multiple test cases.

The first line contains a non-empty string S whose length is not large than 100,000.

The next line contains an integer N (1 <= N <= 10) indicating the number of the short string.

The next N lines, each line contains a short non-empty string whose length is not large than 10.

All strings in the input would not have blank and all characters are lower case.

Output:

For each test case, output a string first indicating the "black hand", and then output an integer indicating the minimal number of the operation. if there are more than one "black hand", please output the smallest one in lexicographical order.

Sample Input:

aaabbbb
2
alice
bob

Sample Output:

bob 1

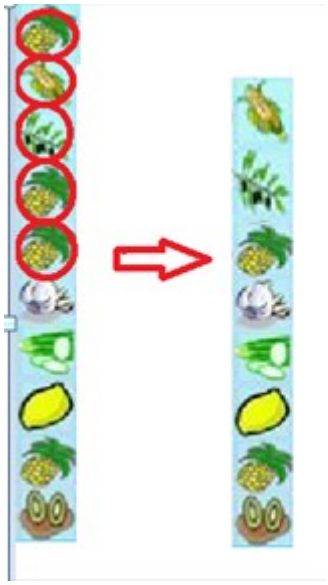
1006

LianLianKan

Description:

I like playing game with my friends, although sometimes look pretty naive. Today I invent a new game called LianLianKan. The game is about playing on a number stack.

Now we have a number stack, and we should link and pop the same element pairs from top to bottom. Each time, you can just link the top element with the same-value element. After pop them from stack, all left elements will fall down. Although the game seems to be interesting, it's really naive indeed.



To prove I am wisdom among my friends, I add an additional rule to the game: for each top element, it can just link with the same-value element whose distance is less than 5.

Before the game, I want to check whether I have a solution to pop all elements in the stack.

Input:

There are multiple test cases.

The first line is an integer N indicating the number of elements in the stack initially. ($1 <= N <= 1000$)

The next line contains N integers a_i indicating the elements from bottom to top. ($0 <= a_i <= 2{,}000{,}000{,}000$)

Output:

For each test case, output “1” if I can pop all elements; otherwise output “0”.

Sample Input:

2

1 1

3

1 1 1

2

1000000 1

Sample Output:

1

0

0

1007

Rescue

Description:

I work at NASA outer space rescue team which needs much courage and patient. In daily life, I always receive a lot of mission, and I must complete it right now.

Today, team leader announced me that there is a huge spaceship dropping anchor in the out space, and we should reach there for rescue. As a working principle, at first, we should check whether there are persons living in the spaceship. So we carry a kind of machine called life sensor which can sense the life phenomenon when the distance between the machine and the living is not farther than the sense radius.

I have read the designing paper of the spaceship in advance. It has a form of a convex polyhedron, and we can assume it is isodense. For best control, control center of the whole ship is located at the center of the mass. It is sure that if someone is still alive, he will stay at the control center.

It's unfortunately that I find the door is stocked when I try to enter into the spaceship, so I can only sense the living out of the space ship. Now I have opened the machine and it's time to set the sense radius of it. I wonder the minimal radius of the machine which can allowe me to check whether there are persons living in the spaceship.

Input:

There are multiple test cases.

The first line contains an integer n indicating the number of vertices of the polyhedron. (4 <= n <= 100)

Each of the next n lines contains three integers xi, yi, zi, the coordinates of the polyhedron vertices (-10,000 <= xi, yi, zi <= 10,000).

It guaranteed that the given points are vertices of the convex polyhedron, and the polyhedron is non-degenerate.

Output:

For each test case, output a float number indicating the minimal radius of the machine. Your answer should accurate up to 0.001.

Sample Input:

```
4
0 0 0
1 0 0
0 1 0
0 0 1

8
0 0 0
```

0 0 2
0 2 0
0 2 2
2 0 0
2 0 2
2 2 0
2 2 2

Sample Output:

0.144
1.000

1008

Spy's Work

Description:

I'm a manager of a large trading company, called ACM, and responsible for the market research. Recently, another trading company, called ICPC, is set up suddenly. It's obvious that we are the competitor to each other now!

To get some information about ICPC, I have learned a lot about it. ICPC has N staffs now (numbered from 1 to N, and boss is 1), and anybody has at most one superior. To increase the efficiency of the whole company, the company contains N departments and the ith department is led by the ith staff. All subordinates of the ith staff are also belong to the ith department.

Last week, we hire a spy stealing into ICPC to get some information about salaries of staffs. Not getting the detail about each one, the spy only gets some information about some departments: the sum of the salaries of staff s working for the ith department is less than (more than or equal to) w. Although the some inaccurate information, we can also get some important intelligence from it.

Now I only concerned about whether the spy is telling a lie to us, that is to say, there will be some conflicts in the information. So I invite you, the talented programmer, to help me check the correction of the information. Pay attention, my dear friend, each staff of ICPC will always get a salary even if it just 1 dollar!

Input:

There are multiple test cases.

The first line is an integer N. (1 <= N <= 10,000)

Each line i from 2 to N lines contains an integer x indicating the xth staff is the ith staff's superior(x<i).

The next line contains an integer M indicating the number of information from spy. (1 <= M <= 10,000)

The next M lines have the form like (x **< (> or =)** w), indicating the sum of the xth department is less than(more than or equal to) w (1 <= w <=100,000,000)

Output:

For each test case, output "True" if the information has no confliction; otherwise output "Lie".

Sample Input:

5
1
1
3
3
3
1 < 6
3 = 4

2 = 2

5
1
1
3
3
3
1 > 5
3 = 4
2 = 2

Sample Output:

Lie
True

1009

Color the Tree

Description:

Now you have a tree with N vertices, and M pens with different color. You want to paint the vertices by your pens, and wondered that how many kinds of different colored trees could be resulted after painting.

Pay attention that two isomorphic trees having same color in corresponding vertices could be considered as two same colored trees.

Input:

There are multiple test cases.

The first line contains two integers N and M. (1 <= N<=50,000,1<= M <= 100,000)

The next N - 1 lines each line contains two integer A_i and B_i indicating there is one tree edge between A_i and B_i. (1 <= A_i, B_i <= N)

Output:

For each test case, output a number module 1000000007 (1e9 + 7) indicating the answer.

Sample Input:

```
1 3

2 3
1 2

5 2
1 2
1 3
3 4
3 5
```

Sample Output:

```
3
6
24
```

1010

The Ghost Blows Light

Description:

My name is Hu Bayi, robing an ancient tomb in Tibet. The tomb consists of N rooms (numbered from 1 to N) which are connected by some roads (pass each road should cost some time). There is exactly one route between any two rooms, and each room contains some treasures. Now I am located at the 1st room and the exit is located at the Nth room.

Suddenly, alert occurred! The tomb will topple down in T minutes, and I should reach exit room in T minutes. Human beings die in pursuit of wealth, and birds die in pursuit of food! Although it is life-threatening time, I also want to get treasure out as much as possible. Now I wonder the maximum number of treasures I can take out in T minutes.

Input:

There are multiple test cases.

The first line contains two integer N and T. (1 <= n <= 100, 0 <= T <= 500)

Each of the next N - 1 lines contains three integers a, b, and t indicating there is a road between a and b which costs t minutes. (1<=a<=n, 1<=b<=n, a!=b, 0 <= t <= 100)

The last line contains N integers, which A_i indicating the number of treasure in the ith room. (0 <= A_i <= 100)

Output:

For each test case, output an integer indicating the maximum number of treasures I can take out in T minutes; if I cannot get out of the tomb, please output "Human beings die in pursuit of wealth, and birds die in pursuit of food!".

Sample Input:

```
5 10
1 2 2
2 3 2
2 5 3
3 4 3
1 2 3 4 5
```

Sample Output:

```
11
```

1011

USACO ORZ

Description:

Like everyone, cows enjoy variety. Their current fancy is new shapes for pastures. The old rectangular shapes are out of favor; new geometries are the favorite.

I. M. Hei, the lead cow pasture architect, is in charge of creating a triangular pasture surrounded by nice white fence rails. She is supplied with N fence segments and must arrange them into a triangular pasture. Ms. Hei must use all the rails to create three sides of non-zero length. Calculating the number of different kinds of pastures, she can build that enclosed with all fence segments.

Two pastures look different if at least one side of both pastures has different lengths, and each pasture should not be degeneration.

Input:

The first line is an integer T(T<=15) indicating the number of test cases.

The first line of each test case contains an integer N. (1 <= N <= 15)

The next line contains N integers li indicating the length of each fence segment. (1 <= li <= 10000)

Output:

For each test case, output one integer indicating the number of different pastures.

Sample Input:

1

3

2 3 4

Sample Output:

1