Problem A: Time Limit Exceeded

As you should be aware by now, one of the verdicts you can get when submitting a solution to a problem is Time Limit Exceeded (TLE). This means that the running time of your solution exceeds the time limit set by judges.

Let us assume that the judging server can make 100,000,000 operations per second. Given the time complexity of your solution expressed using big-O notation, maximum size of the input per test case N, the number of test cases T and the time limit for all cases L can your solution run in time?

Assume that your solution uses only simple operations and disregard any other overhead (e.g. I/O).

Input Format

Input starts with a line containing an integer C (1 <= C <= 100), the number of test cases. C lines follow, each of the format

```
<time_complexity> N T L
```

where N, T and L are integers as described in the problem statement (1 <= N <= 1,000,000, 1 <= T,L <= 10). and *<time_complexity>* is one of the following:

```
O(N), O(N^2), O(N^3), O(2^N), O(N!)
```

Note: We use a very simplified complexity model here and familiarity with big-O notation is not required (or may even be detrimental). Just assume that applying N to the function in the parenthesis is what gives you the total number of operations your solution will use.

Output Format

For each test case print on one line either "TLE!" if the running time of the solution exceeds the time limit for that test case or "May Pass." if it does not.

Sample Input

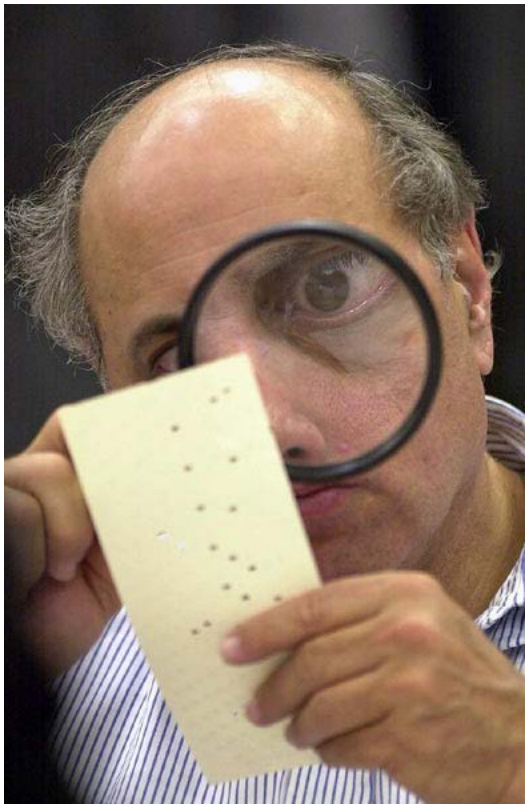
```
5
O(N) 1000 10 10
O(2^N) 1000 10 10
O(N!) 2 10 10
O(N^3) 1000 1 10
O(N^3) 1001 1 10
```

Sample Output

```
May Pass.
TLE!
May Pass.
May Pass.
TLE!
```

Hichem Zakaria Aichour
CCPC 2013

Problem B: Counting Again?



Given an integer N, in how many ways you can tile a 4 x N rectangle with 3 x 1 tiles? Because the answer can be large, we are interested in the remainder after answer is divided by 1,000,000,007.

Input Format

First line of the input contains an integer T (1 <= T <= 100) - the number of test cases. Each test case consist of a single line containing an integer N (1 <= N <= 10,000).

Output Format

For each test case print the the remainder after answer is divided by 1,000,000,007.

Sample Input

```
3
3
6
3333
```

Sample Output

```
3
13
524313417
```

Darko Aleksic
CCPC 2013

Problem C: Coin Turning Game

Alice and Bob are playing the following game (Alice plays first):

They start with an arrangement of N coins in a row (some of them showing heads and some of them showing tails) At each player's turn, they can flip any number of consecutive coins but the rightmost one has to go from head to tail. Whoever is unable to make a move loses.

Given the initial arrangement of coins, if both Alice and Bob play optimally, can Alice win the game?

Input Format

First line of the input contains an integer T (1 <= T <= 100) - the number of test cases. Each test case consist of a single line containing a string S (3 <= |S| <= 15) - the initial arrangement of coins. Each character of S will be either 'H' or 'T' (telling us if heads or tails are up).

Output Format

For each test case determine if Alice can win the game if both she and Bob play optimally. If she can, print "YES" followed by two integers - the position of the rightmost coin flipped (1-based) and the number of coins flipped. If there are several initial moves with which Alice wins the game, you can print any of them. If Alice cannot win the game, just print "NO" for that test case.

Sample Input

```
3
TTTT
HTHT
HTTH
```

Sample Output

```
NO
NO
YES 4 2
```

Darko Aleksic
CCPC 2013

Problem D: Solitaire

You have a deck of N cards valued from 1 to N. The game starts with cards facing down in the "initial" location. There are also three other locations where you can play your cards face up (once they are face up at the top of any of the other piles): "goal", "helper" and "pile". You win the game once all the cards are placed on the goal in ascending order (N on the top). Rules:

1. you can play cards onto "goal" only if the top "goal" card's value is one less than the value of the card you are trying to place (if "goal" is empty, you can only play the card with value 1(one) there). E.g. if the top card in "goal" is 3, you can only play 4 to "goal".
2. you can play cards onto "helper" only if the top "helper" card's value is one more than the value of the card you are trying to place (if "helper" is empty, you can only play the card with value N there). E.g. if the top card in "helper" is 8, you can only play 7 to "helper".
3. you can only move a card onto the "pile" from the top of the "initial" deck by turning that card over(remember? cards in the initial deck are facing down).
4. once all cards are facing up ("initial" deck is empty) and if the game is not finished yet, take the cards from the "pile" and turn all of them over onto the "initial" position. This will be your new "initial deck". To clarify - the top card in "pile" (facing up) will become the bottom card in "initial" (facing down)

What is the minimum number of type 4 moves you need to finish the game?

Input Format

First line of the input contains an integer T (1 <= T <= 100) - the number of test cases. Each test case consists of two lines:

First line contains an integer N (1 <= N <= 1000)
Second line contains description of the "initial" deck. The first number is the value of the card at the bottom of the initial deck facing down (so the last card will be played first onto "pile"). This will be a permutation of the list of integers from 1 to N.

Output Format

For each test case print the minimum number of type 4 moves you neeed to "win" the game on a separate line.

Sample Input

```
2
3
1 2 3
6
6 2 4 3 5 1
```

Sample Output

```
0
1
```

Darko Aleksic
CCPC 2013

Problem E: Matrix

After several years of searching, Morpheus finally found The One - Thomas A. Anderson, a computer programmer known to his friends as Neo. You and Neo are best friends. In fact, you participated several times together in Regional Programming Contests. Morpheus decided to save both of you from the Matrix world by offering you the red pill. After several months of training aboard Morpheus ship, Nebuchadnezzar, it was the time for Neo to see Oracle, who predicted the eventual emergence of The One. While Neo and Morpheus went to see her in the Matrix world you stayed back in the real world to ensure that everything was going according to plan.

After seeing Oracle, Neo and Morpheus were trying to get back to the real world through one of the telephones in the Matrix world. After Morpheus managed to get back and before Neo could use the telephone, an agent destroyed it leaving Neo stranded in the Matrix world. Since Neo was still not at his full potential, his only choice was to avoid fighting agents by running to the nearest telephone that he could reach without the possibility of finding one of the agents on the way.

Since you are the best programmer aboard the Nebuchadnezzar, Morpheus asks you to write a program to determine whether Neo will have to fight an agent or not (since you don't know how the agents will move, your code should assume the worst case scenario). If Neo is able to avoid agents, your program should determine when Neo will get out of the Matrix world given locations of the agents and telephones and the amount of time needed to move between different locations.

Note that if both Neo and an agent arrive at a telephone location at the same time, Neo will have to fight the agent before picking up the phone.

Input Format

First line of the input contains an integer T (1 <= T <= 100) - the number of test cases.

Each test case start with a line containing four integers: N (3 <= N <= 1000) - representing the number of locations in the Matrix world, M - representing the number of paths connecting different locations, NA - representing the number of agents and NT - representing the number of telephones. (NA,NT > 0, NA+NT < N-1)

Next M lines each contain three integers u, v and m (0 <= u,v < N, 1 <= m <= 30). The first two integers represent two locations in the matrix (With 0 representing Neo's starting location) and the third integer represents the time (in minutes) required to between these two locations.

Next line contains NA integers giving the locations of the agents.

The last line of the test case contains NT integers giving the locations of the telephones

Note that there is no agent, nor telephone at location 0 (i.e. Neo's initial location). Also, there is no location that initially has both a telephone and an agent present.

Output Format

For each test case, print "Neo may fight an Agent" if there is no path that ensures the safety of Neo. Otherwise, print the time (in minutes) needed in order to get out of the Matrix.

Sample Input

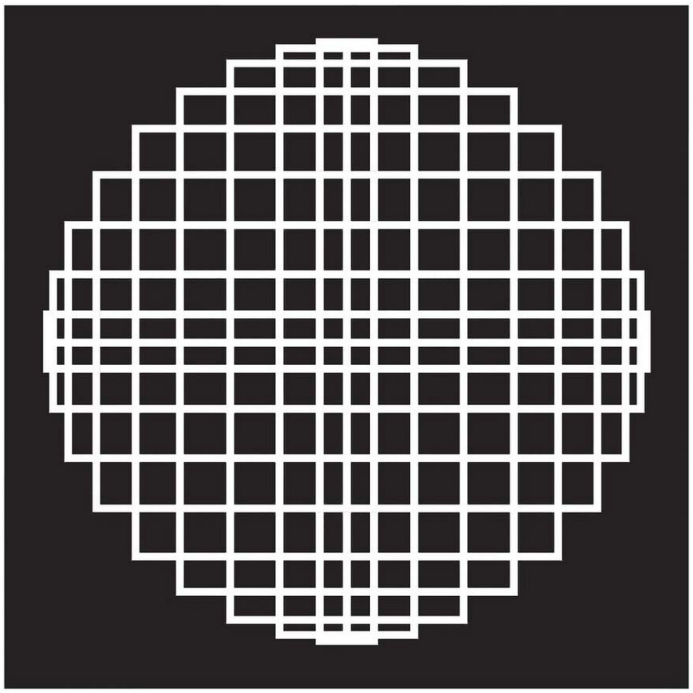
```
2
5 4 1 1
0 1 1
1 2 1
2 3 1
3 4 1
4
1
5 4 1 1
0 1 1
1 2 1
2 3 1
3 4 1
4
2
```

Sample Output

```
1
Neo may fight an Agent
```

Hichem Zakaria Aichour
CCPC 2013

Problem F: Largest inscribed rectangle



Output the maximum area of a rectangle that can be inscribed into a circle of radius R. Rectangle's shorter side should not be longer than B.

Input Format

First line of the input contains an integer T (1 <= T <= 1000) - the number of test cases. Next T lines each contain two integers R and B (1 <= R,B <= 10000).

Output Format

For each test case print on a separate line the maximum area of a rectangle with a maximum shorter side B that can be inscribed into a circle of radius R. Your answer should be rounded to three digits after the decimal point (see sample output).

Sample Input

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

Sample Output

```
1.732
2.000
6.928
```

Hichem Zakaria Aichour
CCPC 2013

Problem G: XOR Sum

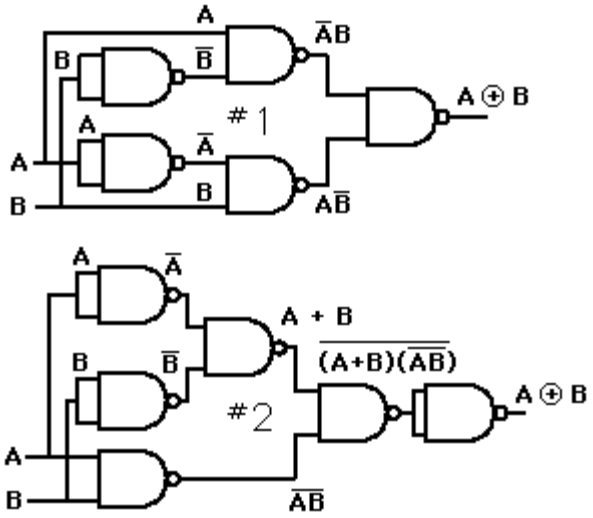
You will be given a list of Q (1 <= Q <= 100,000) instructions.

If the instructions is to "insert N", insert N into the list of numbers (there may be duplicates).

If the instruction is to "print" - print the XOR sum of the largest K (1<=K<=Q) elements in the list (or, if the list contains less than K elements, the XOR sum of all elements in the list).

XOR sum of a list of numbers is the result of XOR-ing all of them. XOR can be applied to two integers using the built in ^ operator in most languages (or xor in Haskell).

Note that XOR function has some useful properties, among them that if N^M=X then N=X^M and M=X^N.



Input Format

First line of the input contains an integer T (1 <= T <= 30) - the number of test cases. Each test case start with a line containing two integers Q and K (1 <= Q,K <= 100,000). Following are Q lines containing one instruction each.

Instructions are in either of the following two forms:

```
insert N
or
print
```

N is a non-negative integer less than 2^31.

Output Format

For each print statement output the sum of (at most) K largest elements in the current list. Note that the list is cleared between test cases.

Sample Input

```
1
5 2
insert 1
insert 2
print
insert 3
print
```

Sample Output

```
3
1
```

Hichem Zakaria Aichour
CCPC 2013

Problem H: There is no place like 127.0.0.1

A friend of yours is a bit on a geeky side. Among other things, he likes to replace some common English words with... IPv4 addresses?!?

Being a good friend you would like to at least remove these literary abominations from his documents. You somehow got hold of his map of words to addresses (or, it turns out, to range of addresses). Now you would like to replace IPv4 addresses with the original English words.

IPv4 address, if you are not familiar with it, is a 32-bit unsigned integer usually represented in dot-decimal notation (e.g. 172.30.12.255 - 4 8-bit unsigned integers separated by dots).

Input Format

First line of the input contains an integer T (1 <= T <= 50) - the number of test cases.

Each test case consists of two sections - map and text.

Map section starts with an integer M (1 <= M <= 1000) on the next line, the number of map entries. Each of the next M lines contains an entry that can contain either word followed by an IP address or a word followed by 2 IP addresses (see sample input). In the first case, only the given address should be replaced with a word. In the second case, any address within the range should be replaced with the given word. If a range is given, the first address will always be less than the second one.

There will be no duplicate entries in the map, meaning no word will apear more than once and no address will repeat. There will be no overlapping ranges of addresses either.

Text section starts with an integer N (1 <= N <= 1000) on the next line, the number of lines of text. Each of the next T lines contains at most 20 words and/or IP addresses separated by single spaces. Words will contain only letters of English alphabet and IP addresses will be valid (consisting only of digits and dots).

Output Format

For each test case and for each line of text, replace all IP addresses for which you have matching words and print the result to the output. If there is no word mapped to an address that appears in the text, just print the address as is.

Sample Input

```
1
3
home 127.0.0.1
smile 255.255.255.0 255.255.255.255
gate 10.0.0.1
4
There is no place like 127.0.0.1
Hannibal at the 10.0.0.1
Let us put a 255.255.255.89 on that face
I am very very 1.2.3.4
```

Sample Output

```
There is no place like home
Hannibal at the gate
Let us put a smile on that face
I am very very 1.2.3.4
```

Hichem Zakaria Aichour
CCPC 2013