programación creativa 2013

ACM-UCLA Programación Creativa 2013

Maratón Local

5 de Octubre de 2013

Universidad Centroccidental Lisandro Alvarado

Contest session

This problem-set contains 9 problems; the pages are numbered from 1 to 11.

Problem A
All Around the World

Filename: `world.c`, `world.cpp` *or* `world.java`
You must read from standard input and print to standard output.

All around the world,
you've got to spread the word.
Tell them what you heard.
We're gonna make a better day.
– Noel Gallagher

What you may call one of the most useless catchy 10-minutes songs ever. However, there are a lot of problems when you try to trust in what the song says, aside from the expensive flights. The fact of trying to understand somebody else in another language is really hard. And that's why we need translators, and we are going to create one just now.

Yes, that's right. Let's create a translator from English to Swedish, and it's really easy, you just go and intercalate a lot of 'f' characters in each word and you have Swedish. For example Hello World would be Hfeflflfo Wfofrflfd in Swedish. Easy, isn't?

Input

The first line contains an integer T, the number of test cases. Then T lines follows, each with a sentence you need to translate, each line won't have more than 200 characters, these being lowercase letters and symbols.

Output

For each test case write the case number followed with the translation.

Sample Input

```
4
all around the world,
you've got to spread the word
tell them what you heard
we're gonna make a better day.

```

Sample Output

```
Case 1: aflfl afrfofufnfd tfhfe wfofrflfd,
Case 2: yfofu'vfe gfoft tfo sfpfrfefafd tfhfe wfofrfd
Case 3: tfeflfl tfhfefm wfhfaft yfofu hfefafrfd
Case 4: wfe'rfe gfofnfnfa mfafkfe a bfeftftfefr dfafy.

```

Problem B
Bullet Game

Filename: `bullet.c`, `bullet.cpp` *or* `bullet.java`
You must read from standard input and print to standard output.

Russian Roulette! It is a game, a really crazy game, where you play with a gun so you could die!

It consists in taking a shot from a random position of the cylinder of a revolver, if it has a bullet you will get a shot in your head, in other cases you will live a bit more. To choose the random position, the player starts in the first position of the gun and move X spots in the revolver, if X gets to be greater than 7 you continue from the first position and so on. Every gun in this game can contain at most 7 bullets in the cylinder but they are not necessarily full all the time. Also, the player will only attempt one shot.

Input

The first line contains an integer T ($T < 20$), the number of test cases. Each case has two lines, the first contains numbers representing each position of the gun, 1 means it has a bale and 0 in other case. The second line contains the integer X ($0 \le X \le 10^{10}$), the position of the cylinder before taking the shot.

Output

For each test case write "dead!" if the player is dead or "alive!" if he survived. Print it without quotes, and with the case number.

Sample Input

```
3
1 1 0 0 1 1 0
1
1 1 0 0 1 1 0
2
1 1 0 0 1 1 0
8

```

Sample Output

```
Case 1: dead!
Case 2: alive!
Case 3: dead!

```

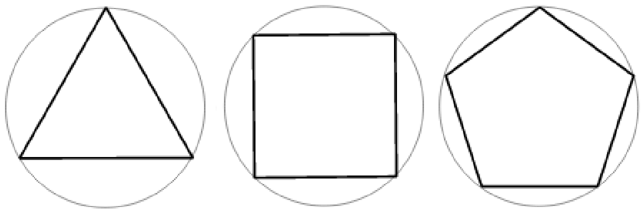
Problem C
Charlie

Filename: `charlie.c`, `charlie.cpp` *or* `charlie.java`
You must read from standard input and print to standard output.

Charlie is a really strange German little girl, she loves maths!!! Can you believe this? But she is not the smarter girl in the world; she needs your help to solve a simple question:

What is the area of one regular convex polygon?

Please think of regular as it has equal sides and angles, and convex as it doesn't have intersections, you can see example of these polygons in the following figures:



Of course, she will give you the data to solve it, remember Charlie is not smart but she is nice, she has the number of sides and the radius of the circumcircle, and with that information it should be enough for you, and remember Charlie is dumb but you are so brilliant.

Input

The first line includes K $(0 < K < 3000)$, the number of the test cases you need to solve, then K lines follows, in each one there will be a test case with two integers indicating the radius r $(0 < r < 20000)$ and the number of sides of the polygon n $(2 < n < 20000)$.

Output

For each line print the area of polygon with 4 decimals. Follow the format of the sample output.

Sample Input

```
4
4 19999
33 4444
2 2000
10 3000

```

Sample Output

```
Case 1: 50.2655
Case 2: 3421.1933
Case 3: 12.5663
Case 4: 314.1590

```

Problem D
Desperate Auction

Filename: `auction.c`, `auction.cpp` *or* `auction.java`
You must read from standard input and print to standard output.

The Venezuelan Government ran out of money and they need more for the upcoming election, so they started a Gold auction, someone that you know that works in the government has hired you and your team to write a tool to help them solve this problem.

The main objective of the auction is to make the most profit from it, but trying to be fair, they want to take in consideration the number of people to sell. So, for this case it has been chosen that a possible solution with a profit p where k people were selected for selling has a score of $p \times 1.2^k$. We want the solution with the higher score.

They have set a number of rules for the action:

- A participant can or not be selected for the selling.
- Only complete gold pieces will be sold, that is, you can't sell half of a piece.
- There are a limited number of pieces of gold to be sold in the auction.
- A participant, if selected, can win at least a 50% of the pieces that he requested. For example, if Nicolas asks for 5 pieces he, if selected, can get a minimum of 3 pieces.
- Any participant called HairGodgiven always wins all the gold for any price.

Input

The input begins with an integer T ($T < 10$) with the number of cases. For each case, there is a line with the number of participants n ($1 \leq n \leq 30$) and the number of pieces to offer in the bid m ($1 \leq m \leq 100$). Then n lines follows with the name of each participant (at most 30 chars, all different), the number of pieces (≤ 100) he or she requested and the price (≤ 1000) he or she offers.

Output

For each case you have to print the score of the better solution possible, given the rules of the problem description, with a precision of 4 digits. See the sample to see the format.

Sample Input

```
2
2 6
Alice 5 20
Bob 3 25
1 20
HairGodgiven 2 1

```

Sample Output

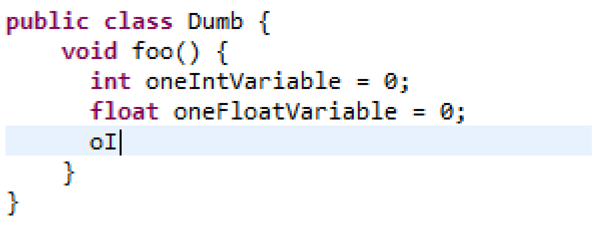
```
Case 1: 194.4000
Case 2: 24.0000

```

Problem E

Eclipse Name Lookup

Filename: `eclipse.c`, `eclipse.cpp` *or* `eclipse.java`
You must read from standard input and print to standard output.

As you may, or not, know, Eclipse has a very useful tool to lookup for symbols when you are writing code, let's say you have the following code,



```
public class Dumb {
    void foo() {
      int oneIntVariable = 0;
      float oneFloatVariable = 0;
      oI
    }
}
```

Assuming the cursor is after the text `oI`, if you press, by default, Ctrl+Space, it will be substituted by the name `oneIntVariable`. However, if the text is only the `o` character, then there would be the tie between the two variables, so it would show a list containing the two variables (Both cases assuming that there aren't other symbols in scope)

The algorithm treats each symbol as a list of consecutive words, where an uppercase character signals the start of a new word. The following list shows examples of how they're separated,

Symbol	Words (separated by commas)
oneIntegerVariable	one, Integer, Variable
oneFloatVariable	one, Float, Variable
TheChosenWord	The, Chosen, Word
IAmAWord	I, Am, A, Word
HTTP	H, T, T, P

To match a given pattern, the algorithm uses the first character as start of the first word, no matter its case. The following uppercase characters are used as the start of the following words. Let's call these characters *starting points*. If there are additional lowercase characters after a starting point, it's supposed they belong to the start of the corresponding word. Given these rules, the patterns *oIV*, *oInV*, *onInVa* all match *oneIntegerVariable*.

Also, the pattern only checks the prefix of a symbol, so the symbol may contain more words than those referenced in the string. For example, *oI* and *oIn* match *oneIntegerVariable*, and *o* match both *oneIntegerVariable* and *oneFloatVariable*.

You are asked to implement this feature. You are given a list of symbols in the scope, and a list of queries, and you are required to count all matches of a query from the original symbols.

Input

The first line of input contains T, the number of test cases. For each case, there is a line with n and q, the number of symbols in the scope, and the number of queries, $1 \leq n, q \leq 200$. After that n lines follow, each with the name of the symbol in the scope, and then m lines, each with the pattern for the query. Each string has a length between 1 and 60 characters and consists only of ASCII alphabetic characters, strings may repeat but are treated independently.

Output

For each case print the case number, followed by q lines, each with the result of each query. See the sample output for the format.

Sample input

```
2
4 5
oneIntegerVariable
oneFloatVariable
oneTime
oneToad
oIn
oI
o
oT
oTi
2 1
hello
world
function
2 1
Hello
Hello
H

```

Sample output

```
Case 1:
1
1
4
2
1
Case 2:
0
Case 3:
2

```

Problem F

Frightening Mitosis

Filename: `mitosis.c`, `mitosis.cpp` *or* `mitosis.java`
You must read from standard input and print to standard output.

There are bacteria that reproduce by mitosis, they do not need a couple, can just duplicate themselves.

Niko is the first of a new kind of bacteria that can reproduces daily, so the first day we have one bacterium (only Niko), at second day we will have two bacteria, at third day we will have four bacteria and so on.

This kind of bacteria is danger because its level of stupidity is very high, so we must destroy them. The first step is to know how many bacteria we will have at day N.

Input

The first line contains the number of test cases, then for each line there is a case with an integer N ($1 \leq N \leq 60$).

Output

For each test case print a line with the number of bacteria. Follow sample output format.

Sample input

```
3
1
2
3

```

Sample output

```
Case 1: 1
Case 2: 2
Case 3: 4

```

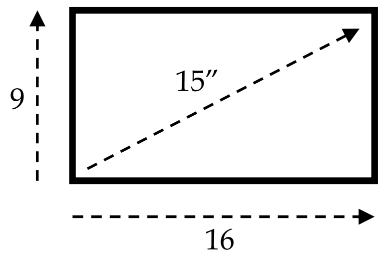
Problem H

Half the Screen Away

Filename: `screen.c`, `screen.cpp` *or* `screen.java`
You must read from standard input and print to standard output.

One is used to hear about 15” flat screens in laptops, 40” TV giant screens, there are even several relations, the old 4:3, or the lately common 16:9, or 2:1 used in movie theatres. It is all very confusing, so let’s try to understand them once for all.

First, companies always give a size for a screen as the length of the diagonal of the device, but for now let’s suppose it is the length of the proper screen. They also give the relation between the width and height, represented in the form $a : b$, which means, for every unit in the height of the device there will be a/b units in the width. For example, look at the figure at the right.



The figure shows an example of 15” screen, with a relation 16:9. Now the problem is, we want to know how big really the screen is, and we think that with the area you give a good idea of that. So, given the length of the diagonal, and the relation, find the screen area.

Input

The input consists of T in the first line, the number of test cases, then T lines follows, each representing a test case, with the integers d, a, b $(1 \leq a, b \leq 20,\ 1 \leq d \leq 60)$. These are the length of the diagonal, and the relation $a : b$ for the screen.

Output

For each case print one line with area of the screen, in inches, with a precision of 2 digits. See the sample output for the format.

Sample input

```
4
15 16 9
15 4 3
40 2 1
40 1 2

```

Sample output

```
Case 1: 96.14
Case 2: 108.00
Case 3: 640.00
Case 4: 640.00

```

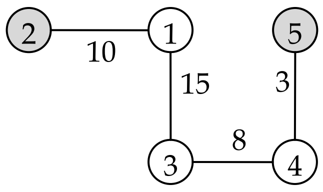
Problem I

In Case of Fire

Filename: `fire.c`, `fire.cpp` *or* `fire.java`

You must read from standard input and print to standard output.

Look, we are looking to expand our business and we need to think about how to prepare our site in the case of that a fire occurs, for that we took our new building drafts and from them defined the intersections and edges. We defined several of those intersections as fire exits, and we need to make sure that from every intersection in the map the distance to a fire exit is no more than a specified value, t.



For example, in the figure shows two fire exit locations, 2 and 5, and the labels attached to the edges are their lengths. Also, given $t = 14$, we see that from every node you can get to a fire exit without breaking the limit, not so with $t = 10$.

As you are our new employee we need to ask you to build a solution for our problem. And don't worry about JJ, I know he worked in other two companies whose offices burned down, but it wasn't his fault. He told us that they had bad fire handling techniques, that's why he is even helping us checking at the building drafts!

Input

The input consists of T in the first line, the number of test cases, then T lines follows, each with n, m, k and t $(n \leq 100, k \leq n, m \leq n(n-1)/2, t \leq 10^6)$, the number of nodes, of edges, of fire exits, and the threshold. The next line includes k integers, the index of each fire exit. Then, m lines follows representing each edge, with u, v and w $(1 \leq u, v \leq n$, $\mathrm{u} \neq \mathrm{v}, 1 \leq \mathrm{w} \leq 1000)$, where u and v are vertices of the edge and w its length, all integers.

Output

For each case print one line that says "YES" if you can to a fire exit from every node in the map or "NO" otherwise. See the sample output for the format.

Sample input

```
1
5 4 2 14
2 5
1 2 10
1 3 15
3 4 8
4 5 3

```

Sample output

```
Case 1: YES

```

Problem K

Krogan Anger Management Program

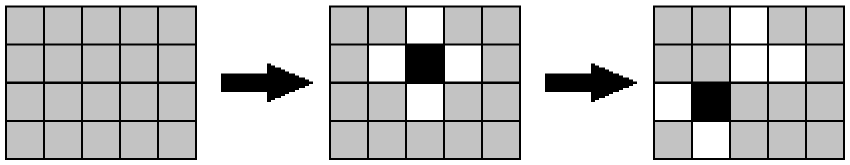
Filename: `krogan.c`, `krogan.cpp` *or* `krogan.java`

You must read from standard input and print to standard output.

First, if you haven't played Mass Effect then go and do it, this contest is not that important. Now, the Krogans are a sapient race from the Mass Effect universe, think of them as bad-tempered humanoid sharks. Well, we are going to propose an Anger Management Program for them!

See, we think that the first step is to make them learn patience, and for that, nothing works better to make someone concentrate than a puzzle! Let's take one from their sister game, Dragon Age. The mechanics are as follows. You have a board with n rows and m columns, where each intersection is a cell. All cells are facing down at the start, but when you press one of the cells, both that cell and the adjacent ones are switched to face the opposite direction. That is, if a cell is facing up, then after pressed it will face down, and vice versa. Two cells are adjacent if they share one edge. The goal is to make every cell to be facing up.

In the following example, you can see it in action. Cells facing down are gray, cells facing up are white, and the pressed cell is black.



We need you to find a solution for several boards. In the case that the Krogans can't solve any of them, and we aren't able to show them that it's possible, then they will probably kill us all. So, help, please.

Input

There is a line with T ($T < 100$), the number of test cases, followed by T lines, each one describing a test case, each one contains two positive integers, n and m. The number of cells will never be greater than 20.

Output

For each case print one line with the solution for the test. The solution is represented by the cells it must be pressed, and for that, each cell is labelled following the next procedure, the cells in the first row are labelled as $1,2,3,\dots,m$; the cells of the second row are labelled $m+1, m+2, \dots, 2m$, and so on.

The output must include the case number, as shown by the sample output, and each label must be separated by exactly one whitespace character.

For each case there may be several solutions, so choose the one with minimum presses, and from them choose the one who comes lexicographically first. That is, choose the one that has the minimum first element, and if there is still a tie, then choose the one with the minimum second element, and so on.

Sample input

```
4
1 1
3 2
2 8
4 5

```

Sample output

```
Case 1: 1
Case 2: 1 6
Case 3: 2 3 6 7 10 11 14 15
Case 4: 2 3 4 7 9 12 14 17 18 19

```