The 34th ACM International Collegiate Programming Contest Asia Regional Contest - Hefei

Problem Set

Hosted by University of Science and Technology of China

Oct. 11, 2009

This problem set contains 10 problems; 23 pages totally.

Problem A: Airport

Martin is the conductor of an airport in Mars. As the Martian scientists invented a new kind of aircraft, Martin is going to have a nightmare.

The new aircraft looks somewhat like a sphere with different radii according to the model. With the powerful engine, the aircraft can accelerate itself to a certain speed in no time, and travel in that speed and direction until stopped. Furthermore, its amazing braking system is able to stop the aircraft immediately.

Unfortunately, due to the global financial crisis, the Martian can't afford the navigation system of the aircraft. That leads to a serious problem: the aircrafts may hit each other while travelling.

Martin's airport is going to replace the old aircraft with the new ones. If any collision happens, Martin will lose his job. Martin has got the speed and radius of every aircraft, together with their travelling plans. He wants to know when the first collision happens, so as to take some actions to avoid the accident. You are the most brilliant programmer in Mars, please help Martin!

Input

The input consists of multiple test cases.

In each case, the first line consists of an integer n ($0 \leqslant n \leqslant 1000$), which is the number of aircrafts. Each of the next n lines describes an aircraft. The description for each aircraft consists of 9 integers separated by single space: x_i y_i z_i x_i' y_i' z_i' r_i t_i v_i. (x_i, y_i, z_i) and (x_i', y_i', z_i') indicate the initial position and destination of the center of i^{th} aircraft. r_i, t_i and v_i indicate the radius, starting time and speed of the aircraft. The aircraft will stay where it was when stopped. It is guaranteed that no two aircrafts will touch or overlap with the other initially, and absolute values of all integers will not exceed 10000. Notice that two aircrafts touching each other is considered to be collision.

The last line of input is '-1', which denotes the end of input file.

Output

For each case, output when the first collision happens with the precision of 0.01 in a single line. If there will never be a collision, just output the word "Never".

Sample Input

```
2
0 0 0 10 0 0 1 0 1
10 2 0 0 2 0 1 0 1
```

```
2
1 1 1 3 3 3 2 10 2
-1 -1 -1 -3 -3 -3 1 2 1
4
0 0 0 100 50 20 1 1 2
100 50 20 0 0 0 2 3 4
200 300 150 100 50 20 3 0 1
10 10 10 110 60 30 1 1 2
2
0 0 0 4 5 3 3 0 1
6 7 5 1 2 0 2 5 1
1
1 2 3 1 2 3 100 10 8
-1
```

Sample Output

```
5.00
Never
20.76
5.26
Never
```

Problem B: Mersenne Prime

A Mersenne number is a positive integer that is one less than a power of two:

$$M_n=2^n-1.$$

A Mersenne prime is a Mersenne number that is prime. As of June 2009, only 47 Mersenne primes are known; the largest known prime number ($2^{43,112,609} - 1$) is a Mersenne prime.

They are named after 17th century French scholar Marin Mersenne(1588 – 1648), who compiled a list of Mersenne primes with exponents up to 257. His list was only partially correct. Mersenne gave little indication how he came up with his list, and its rigorous verification was completed more than two centuries later. Many greatest mathematicians made contribution on Mersenne primes, including Euler.

Now you have more computing power than those great mathematicians. Given a positive integer n (n < 258), you should tell whether the M_n is a prime number. The input ends with a 0.

Input

Each line of the input is a test case, which contains a positive integer n, n < 258.

The last line of input is “0”, which denotes the end of input.

Output

For each case, output the number of n. After that, output “Prime” if the corresponding Mersenne number is prime number, otherwise output “NotPrime”..

Sample Input

```
2
66
67
127
0
```

Sample Output

2:Prime
66:NotPrime
67:NotPrime
127:Prime

Problem C: Bubble Breaker

Bubble breaker is a popular game on mobile phone. It is simple but interesting. The gameboard consists of a screen of differently-colored bubbles arranged in a matrix. There are five different colors: red, blue, green, yellow and purple. The player then clicks on any two or more connecting same-colored bubbles to eliminate them from the matrix, earning an appropriate amount of points in the process. The more bubbles eliminated one time, the higher the points added to the player's score. More specifically, the rules of the game are as follows:

1). Bubbles are 4-connected to their horizontal and vertical neighbors. Two or more connecting bubbles with the same color can be broken at once;

2). Once some bubbles are broken, other bubbles on top of them fall downwards. Whenever the player clears an entire bubble column, the columns on its left slide right and fill the gap.

3). The scoring of each bubble-breaking operation can be expressed in the formula "Y=X(X-1)". X represents the amount of bubbles broken together, Y is the resulting score. For example an elimination of 16 bubbles will result in 240 points (240=16(16-1)).

John likes the game very much. And he has worked out a simple strategy to win high scores for most cases. In his strategy, John assigns different priorities to bubbles with different colors. If there are more than one set of connected bubbles with different colors that can be broken in the gameboard, John always breaks red bubbles at first, and then green bubbles, then blue bubbles, then yellow bubbles, and finally purple bubbles. If there are multiple choices with the same color, John can always break the proper bubbles and achieve a final score as high as possible. Now the problem is, given a new bubble breaker game, what final score can John win using his strategy?

Input

The input consists of multiple test cases.

Each test case starts with a line containing two integers ***N*** and ***M*** (1<=***N***<=16, 1<=***M***<=11), which are the number of rows and columns of the gameboard. Each of the following ***N*** lines contains *M* integers, ranging from 0 to 4, representing the bubbles in the gameboard with different colors (0 = red, 1 = green, 2 = blue, 3 = yellow, 4 = purple).

The last test case is followed by a line containing two zeros.

Output

For each test case, print a line containing the test case number (beginning from 1) followed by an integer which is the final score that John will get.

Sample Input

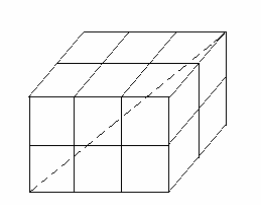
```
4 3
0 1 2
1 2 3
3 4 0
1 0 0
4 4
0 1 2 3
1 2 3 0
2 3 0 1
3 0 1 2
0 0
```

Sample Output

```
Case 1: 10
Case 2: 0
```

Problem D: Laser in Cuboids

As we know, a cuboid (rectangular solid) has three mutually perpendicular edges: a, b, c. When a, b and c are all integer, the cuboid can be regarded as $a \times b \times c$ small cubes gluing together; each of the small cube is 1×1×1.



Now, as in the above figure, there is a laser beam from one vertex to its diagonal opposite vertex. The width of the laser is negligible, but we care about how many small cubes the laser passes through.

Given the integer length of the three edges of the cuboid, your job is to work out how many small cubes the laser passes through (not just contact with its edges or vertices).

Input

Each line of the input is a test case, which contains three integer a, b and c, $0< a, b, c<10^6$. The last line of input is "0 0 0".

Output

For each case, output the number of small cubes the laser passing through.

Sample Input

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

Sample Output

```
3
4
3
```

Problem E: Dominating Patterns

The archaeologists are going to decipher a very mysterious “language”. Now, they know many language patterns; each pattern can be treated as a string on English letters (only lower case). As a sub string, these patterns may appear more than one times in a large text string (also only lower case English letters).

What matters most is that which patterns are the dominating patterns. Dominating pattern is the pattern whose appearing times is not less than other patterns.

It is your job to find the dominating pattern(s) and their appearing times.

Input

The entire input contains multi cases. The first line of each case is an integer, which is the number of patterns N, 1<=N<=150. Each of the following N lines contains one pattern, whose length is in range [1, 70]. The rest of the case is one line contains a large string as the text to lookup, whose length is up to 10^6.

At the end of the input file, number ‘0’ indicates the end of input file.

Output

For each of the input cases, output the appearing times of the dominating pattern(s). If there are more than one dominating pattern, output them in separate lines; and keep their input order to the output.

Sample Input

```
2
aba
bab
abababababac
6
beta
alpha
haha
delta
dede
tata
dedeltalphahahahototatalpha
0
```

Sample Output

```
4
aba
2
alpha
haha
```

Problem F: K-neighbor substrings

The Hamming distance between two strings of the same length is defined as the number of positions at which the corresponding characters are different. For example, the Hamming distance between "abbab" and "bbabb" is 3.

A string is called a K-neighbor of another string if and only if they are of the same length and the Hamming distance between them is not larger than K. In this problem, given an integer K and two strings A and B which only contain character 'a' and 'b', you are to count how many different sub-strings of A are K-neighbors of B.

Input

The input consists of multiple test cases. Each test case starts with a line containing one integer K(0<=K<=100,000). The following two lines give two non-empty strings consisting of 'a' and 'b', which are string A and string B, respectively. The length of strings A and B will both lie between 1 and 100,000, inclusive.The last test case is followed by a line containing one -1.

Output

For each test case, print a line containing the test case number(beginning with 1) followed by the number of different sub-strings of string A which are K-neighbors of string B.

Sample Input

```
0
aabbab
ab
1
aabbab
ab
2
aabba
ab
-1
```

Sample Output

```
Case 1: 1
```

Case 2: 3
Case 3: 4

Problem G: New Game

Alice and Bob are fond of a new interesting game. In this game, they play by turns in a Directed Acyclic Graph (DAG). Every node of the graph is assigned a non-negative integer at the beginning. Each round a player must choose a node having a positive integer and transfer 1 to one of its succeeding nodes, i.e. its integer decreases by 1 and certain succeeding node's integer increases by 1. Who cannot find such a node to transfer will lose, i.e. the integer is 0 in every node having at least 1 succeeding node.

Before playing the game, they come to an agreement about the integer assignment. Alice plays first. To be fair, Bob is responsible for assigning an integer to each node. But there is a constraint: the sum of all integers must be a predetermined value that they both agree.

Now, given the sum, Alice wonders how many distinct assignments which will prevent her from winning. Two assignments are different if there is one node assigned different integers in the two assignments.

Alice and Bob are so smart that they can always find the best strategy.

Input

The first line is a number indicating the number of test cases.

The first line of each test case includes three number N, M and S (2<=N<=100, 1<=M<=10000, 1<= S <=10000) indicating the number of nodes, the number of edges and the sum of all integers respectively. Nodes are numbered by 0…N-1.

The following M lines each have two integers, u and v, indicating an edge from u to v (0 <= u, v <N).You can assume there is no cycle in the graph.

Output

For each test case, output a single integer indicating how many distinct assignments in which Alice can NOT win. Because it can be very large, you should output the remainder divided by 1,000,000,007.

Sample Input

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

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

Sample Output

```
2
10
6
```

Problem H: Chinese Paper Cutting

“Paper cutting is the art of cutting paper designs. ”
“Chinese Paper Cutting or Jianzhi is the first type of papercutting design, since paper was invented by Cai Lun in the Eastern Han Dynasty in China. The art form later spread to other parts of the world with different regions adopting their own cultural styles. Because the cut outs are also used to decorate doors and windows, they are sometimes referred to *"chuāng huā"*, meaning Window Flower.”

--WikiPedia

Alice is a fan of interesting Chinese paper cutting (ICPC). When she first saw it in her grandmother’s house, she immediately fell in love with this art. And from then on, she began to collect various paper cutting designs and tried to create her own works as well. At first, she couldn’t manipulate her scissors and often cut inappropriate parts. Sometimes, she even hurt herself. She had many beautiful designs in her mind, but she was unable to realize them. She was so disappointed for that. However, she didn’t give up. She persisted in practicing, learning from books and experts. At last, she found a key feature of paper cutting. That is symmetry.
Symmetry is very important for paper cutting. It not only beautifies the whole work but also makes the cutting process easy. Therefore, cutting each part of a work independently is not a good choice. Instead, it will be better to fold and cut alternately.
Now Alice is experienced and works out a paper cutting process which is safe and easy. So she invents a machine that can perform paper cutting following her instructions.
This machine has a wide foldable board and a coordinate system on the board. Before the machine runs, she puts a piece of rectangular paper on the board. Two edges of the paper accord with X-axis and y-axis respectively. X-axis is from left to right and y-axis is from top down.
Then she starts up the machine and performs a paper cutting process by repeating two types of instructions:

- **`fold A B:`**
  `To execute this instruction, the part of the board containing the point A will be folded onto the other part. So the point A will coincide with B after folding. Then, the folded part will come back to the original position.`

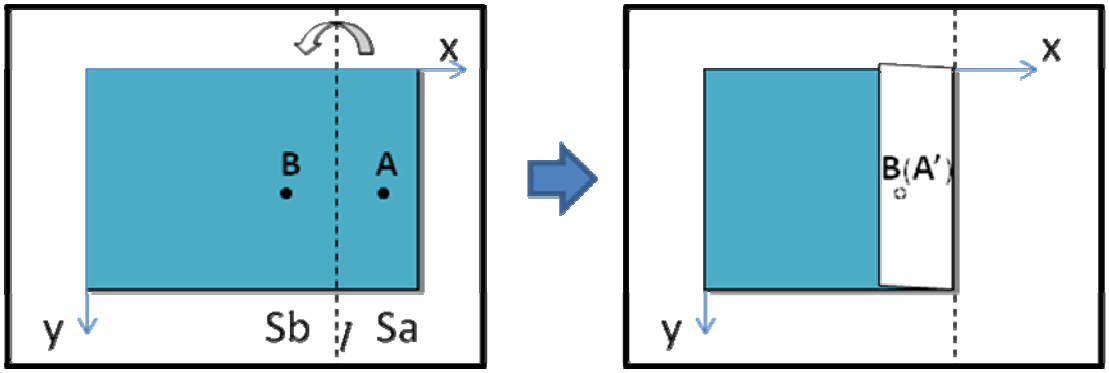


Figure 1

As showed in figure 1, Sa is folded onto Sb to make A and B coincide, meanwhile, the paper is folded along the line l . Then, Sa is put back to the original position. But the folded area of the paper is left on Sb.
In some cases, the paper may be completely within Sa. If so, the whole paper will be reversed and left on Sb. In contrast, the paper will be unchanged if it is completely within Sb.
The folding is limited to be vertical or horizontal.

- **cut** $n\ A_0\ A_1 \ldots A_{n-1}$

  The machine is equipped with a small sharp knife. First, it will use the knife to cut the paper along the line segment A_0A_1. If there are multiple layers under A_0A_1, it will penetrate all of them. After that, it will continue to cut along the segment A_1A_2 and then A_2A_3, etc. A_i can be outside the paper. The machine will cut nothing if it finds the knife is outside the paper.
  Sometimes it may happen to cut along the joining line of two layers. The knife is so sharp that it will cut off them all the same.

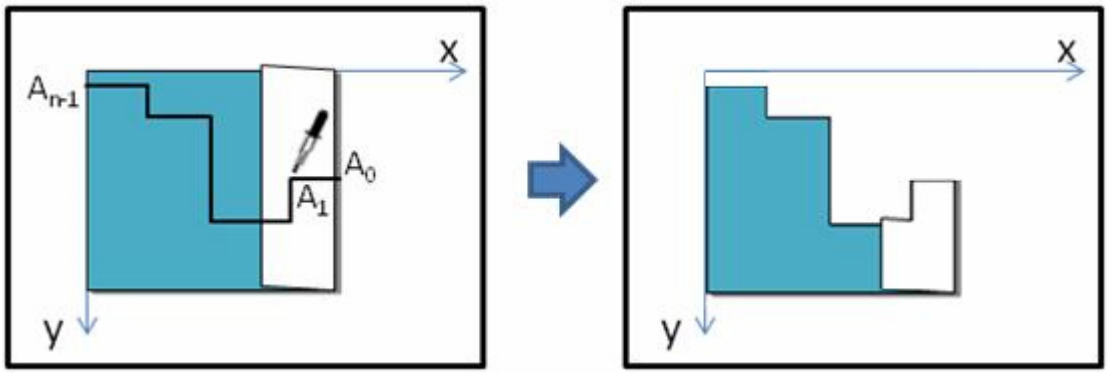


Figure 2

If the paper is divided to several pieces after cutting, the machine will automatically get rid of all pieces but the one with the largest area (after unfolding). If there is a tie, it will keep the one containing the most top-left point in the original paper, i.e. the most left point in the most top horizontal line.

Finally, she unfolds the paper and sees a fantastic design.

For some complex designs, she is not sure whether her instructions will exactly make what she wants. She doesn't want to waste paper. So she needs you to write a program that can simulate her instructions and tell her the final design. Given her instructions, your job is to print the final design.

Input

The first number indicates the number of test cases. It will not exceed 50.

For each test case, the first line contains two integers L, W (0<L, W<=100) indicating the length and width of the paper. The top left corner of the paper is at (0, 0).

The second line is a single number T indicating how many instructions she will execute. (0<= T <= 100)

The following T lines each are an instruction:

$$\mathbf{cut}\ n\ x'_0\ y'_0\ x'_1\ y'_1 \dots x'_{n-1}\ y'_{n-1}$$

Or

$$\mathbf{fold}\ x_0\ y_0\ x_1\ y_1$$

Here $(x_0, y_0), (x_1, y_1)$ are the two points which should be in the same place after folding. You can assume that they are distinct and locate in the same vertical or horizontal line.

$(x'_0\ y'_0)(x'_1\ y'_1) \dots (x'_{n-1}, y'_{n-1})$ indicates the knife's path. All the segments are veridical or horizontal.

$0 \le x_0, x_1, x_i' \le L \quad 0 \le y_0, y_1, y_i' \le W$

$2 \le n \le 10$ x_0, x_1, x_i' y_0, y_1 and y_i' are all integers.

Output

The output of each case should consist of W rows. Each row includes L characters. If the area of [j-1,j]*[i-1,i] remains in the final design, you should print a '@' for the j-th character of the i-th row; Otherwise, print a '.'
Output an empty line at the end for each test case.

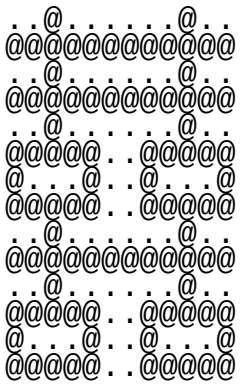
Sample Input

```
3
3 3
3
fold 0 3 0 0
fold 3 0 0 0
cut 3 0 1 1 1 1 0
4 3
3
fold 4 0 0 0
cut 2 2 0 2 3
cut 4 2 1 1 1 1 2 2 2
12 14
8
fold 12 0 0 0
fold 0 14 0 5
fold 0 9 0 4
fold 0 0 0 5
fold 0 3 0 4
fold 0 0 5 0
cut 3 0 6 4 6 4 10
cut 5 6 4 3 4 3 5 5 5 5 10
```

Sample Output

```
.@.
@@@
.@.

@@..
@...
@@..
```



Problem I: Post offices

There is a straight highway with N villages alongside it. The villages are numbered from 1 to N in one direction of the highway.The government is planning to build at most M post offices in some of the villages.

The amount of money to build a post office in the i-th village is C_i and the i-*th* village can be served by any post office within R_i kilometers to the left or right of it. If a village has no post office built and no post offices in other villages can serve it, the government has to compensate the villagers P_i money. Here C_i, R_i and P_i are all non-negative integers.

You are to help the government to find a strategy of minimum cost.

Input

The input consists of multiple test cases. Each test case starts with a line containing two integers N(2<=N<=20,000) and M (1<=M<=N, M<=100).
The following line contains N-1 positive integers, which are the distances of between village 1 and villages 2 ,3,...,N in kilometers.
The distances will be not greater than 1,000,000,000 and strictly increasing.
The third line of each test case contains N integers C_1, C_2, ... , C_N, each of which is between 0 and 10,000, inclusive.
The fourth line of each test case contains N integers R_1,...,R_N, each of which is between 0 and 1,000,000,000, inclusive.
The last line of each test case contains N integers P_1, ... , P_N, each of which is between 0 and 10,000, inclusive.

The last test case is followed by a line containing two zeros.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by the minimum amount of money the government has to pay.

Sample Input

```
3 2
1 2
2 3 2
1 1 0
10 20 30
```

3 2
10 20
100 2 300
5 6 7
10 100 400
0 0

Sample Output

Case 1: 4
Case 2: 312

Problem J: Super squares

There is a small kingdom. The kingdom consists of N towns and each town has some residents.

Some pairs of towns are connected by bidirectional highways such that there is exactly one highway path between each pair of towns. The king wants to select M towns to build super squares in them so that the residents in the kingdom can gather in these super squares to celebrate new years together. For some security reason, the selected M towns must satisfies the condition that starting from any selected town, someone can reach any other selected town by highways without passing through any town which is not selected.

Each highway has a length. People always travel to the nearest town with a super square to celebrate new years.

The king wants to make the total distance people have to travel to celebrate every new year as small as possible. You are to help the king. The total distance is the summation of distance every resident travel.

Input

The input consists of multiple test cases. Each test case starts with a line containing two integers N(2<=N<=2,000) and M (1<=M<=N, M<=500).

The following line contains N positive integers, the i-*th* of which is the number of resident in the i-th town and will not be greater than 1000.

Each of the next N-1 lines contains three integers a, b and c (1<=a<b<=n, 1<=c<=1,000), which means there is a bidirectional highway between the a-*th* town and the b-*th* town and the length of the highway is c kilometers.

The last test case is followed by a line containing two zeros.

Output

For each test case, print a line containing the test case number(beginning with 1) followed by the minimum total distance people has to travel.

Sample Input

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

100 10 100
1 2 1
2 3 1
0 0

Sample Output

Case 1: 13
Case 2: 100