The 33rd ACM International Collegiate Programming Contest Asia Regional Contest - Hefei

Problem Set

Hosted by University of Science and Technology of China

Nov. 16, 2008

This problem set contains 8 problems; 15 pages totally.

Problem A: A Simple Game

Alice and Bob are playing a simple game. They have *N* integer numbers and a target number *T* in common. Either of them independently and randomly picks a number from the *N* numbers. They win the game if the product of the two picked numbers is strictly greater than the target number *T*. You are to calculate the probability that they will win. Assume that each number is picked with the same probability.

Input

The input consists of multiple test cases. Each test case consists of two lines.

The first line contains two integers *N* ($1 \le N \le 30{,}000$) and *T* ($-10^9 \le T \le 10^9$).

The second line contains *N* integers numbers that Alice and Bob have, each of which will be between $-30{,}000$ and $30{,}000$, 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 probability of which Alice and Bob will win the game. The probability is printed as a fraction number formatted as "a/b", where the greatest common divisor of *a* and *b* must be 1.

Sample Input

```
2 0
2 -9
4 5
1 -4 3 -2
0 0
```

Sample Output

```
Case 1: 1/2
Case 2: 1/4
```

Problem B: Discrete Square Roots

A square root of a number *x* is a number *r* such that $r^2 = x$. A discrete square root of a non-negative integer *x* is a non-negative integer *r* such that $r^2 \equiv x \bmod N$, $0 \le r < N$, where *N* is a specific positive integer and mod is the modulo operation.

It is well-known that any positive real number has exactly two square roots, but a non-negative integer may have more than two discrete square roots. For example, for $N = 12$, 1 has four discrete square roots 1, 5, 7 and 11.

Your task is to find all discrete square roots of a given non-negative integer *x*. To make it easier, a known square root *r* of *x* is also given to you.

Input

The input consists of multiple test cases. Each test case contains exactly one line, which gives three integers *x*, *N* and *r*. ($1 \le x < N$, $2 \le N < 1{,}000{,}000{,}000$, $1 \le r < N$).

It is guaranteed that *r* is a discrete square root of *x* modulo *N*. The last test case is followed by a line containing three zeros.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by a list of corresponding discrete square roots, in which all numbers are sorted increasingly..

Sample Input

```
1 12 1
4 15 2
0 0 0
```

Sample Output

```
Case 1: 1 5 7 11
Case 2: 2 7 8 13
```

Problem C: Necklace

A necklace in an undirected graph is a sequence of cycles C_1, C_2, ... , C_k ($k \geq 1$), satisfying the conditions below:

1. Any two cycles have no edges in common.
2. There is exactly one common vertex between two adjacent cycles C_i and C_{i+1} ($1 \leq i < k$)
3. Any two non-adjacent cycles are vertex disjoint, i.e. no vertices in common.

Note that any vertex appears in a cycle at most once.

A necklace between two vertices *S* and *T* is a necklace $C_1, C_2, ..., C_k$ such that *S* belongs to C_1 and *T* belongs to C_k.

Given an undirected graph and two vertices *S* and *T*, you need find whether a necklace between *S* and *T* exists.

Input

The input consists of multiple test cases. Each test case starts with a line containing two integers *N* ($2 \leq N \leq 10{,}000$) and *M* ($1 \leq M \leq 100{,}000$), which are the number of vertices and the number of edges in the undirected graph, respectively.
Each of the following *M* lines contains two integers *A* and *B* ($1 \leq A \neq B \leq N$), which indicates an undirected edge between vertices *A* and *B*. Vertices are numbered from 1 to *N*.
The last line of each test case contains two integers *S* and *T* ($1 \leq S \neq T \leq N$).
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 "YES", if the required necklace exists, otherwise "NO".

Sample Input

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

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

Sample Output

```
Case 1: YES
Case 2: YES
Case 3: NO
```

Problem D: Polynomial-time Reductions

In computational complexity theory, polynomial-time reduction is an important concept.

If the existence of a polynomial-time algorithm for problem *B* implies that problem *A* also has a polynomial-time algorithm, we say that problem *A* has a polynomial-time reduction to problem *B*. The relation of reduction is transitive, i.e. if *A* has a reduction to *B* and *B* has a reduction to *C*, then *A* has a reduction to *C*.

Theoretical computer science researchers have found hundreds of polynomial-time reductions between problems, which build a large network of reductions in computational complexity theory.

In this network, some reductions are explicitly presented and others exist implicitly. For example, if the reductions from *A* to *B* and from *B* to *C* are explicitly presented and the reduction from *A* to *C* is not explicitly presented, we say that the reduction from *A* to *C* exists implicitly.

In fact, some reductions are not necessary to present explicitly in the network. For example, if reductions from *A* to *B* and from *B* to *C* exist explicitly or implicitly in the network, then the reduction from *A* to *C* is not necessary to present explicitly. By this fact, it's possible to simplify the network of reductions.

Your task is just to simplify the network of reductions such that the number of reductions explicitly presented in the simplified network is minimized and reduction from problem *A* to problem *B* exists explicitly or implicitly in the simplified network if and only if it exists in the original network explicitly or implicitly.

Input

The input consists of multiple test cases. Each test case starts with a line containing two integers *N* and *M* ($1 \le N \le 100$, $0 \le M \le 10000$), which are the number of problems and the number of explicitly presented reductions in the network. The problems are numbered from 1 to *N*.
Each of the following *M* lines contains two integers *A* and *B* ($1 \le A \le N$, $1 \le B \le N$, $A \ne B$), which means a polynomial-time reduction from problem *A* to problem *B* is explicitly presented in the network. 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 a integer which is the number of explicitly presented reductions in the simplified network.

Sample Input

```
3 3
1 2
2 3
1 3
4 6
1 2
2 1
2 3
3 2
3 4
4 3
0 0
```

Sample Output

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

Problem E: 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 ith village is C_i and a post office in the ith village can serve all villages within R_i kilometers to the left and right of it.

If the ith 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 with minimum cost.

Input

The input consists of multiple test cases. Each test case starts with a line containing two integers *N* ($2 \le N \le 10000$) and *M* ($1 \le M \le N, M \le 100$).

The following line contains $N-1$ positive integers, which are the distances 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_2, ... , 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_2, ... , P_N, each of which is between 0 and 10,000, inclusive.

The last test case is followed by a line containing one zero.

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: 3
Case 2: 312
```

Problem F: Programmers

In some international companies (e.g. SUN and IBM), programmers can work at home via internet.
Each programmer has his/her own work time interval. For example, Tom always works from 11:15 to 19:34 every day and Alice works from 22:14 to 05:13 of the morrow. Note that the time intervals may span two days, but the lengths of them will be strictly less than 24 hours.
Two programmers can talk with each other by instant messenger software if and only if their work time intervals overlap. Note that only having common beginning or ending point doesn't work. For example, the interval (11:15, 19:34) overlaps with the interval (19:33, 20:10), but (11:15, 19:34) doesn't overlap with (19:34, 11:15).
Now a big project needs as many programmers as possible such that any two of them can talk with each other at some time in their work time intervals.
You are to find the maximum number of programmers who can participate in the project.

Input

The input consists of multiple test cases. Each test case starts with a line containing one integers N ($1 \leq N \leq 1000$), which is the number of programmers in the company. Each of the following N lines gives a work time interval of a programmer in the format of "ab:cd-ef:gh", where "ab:cd" and "ef:gh" are beginning time and ending time written in the 24-hour notation. You can assume that the input times are legal and the beginning and ending times are different.
The last test case is followed by a line containing one zero.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by a integer which is the maximum number of programmer who can participate in the project.

Sample Input

```
3
21:59-00:43
00:42-13:03
12:00-22:00
3
21:59-21:58
```

```
21:58-21:59
21:58-05:08
0
```

Sample Output

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

Problem G: Rational Number Approximation

It is well-known that any rational number can be represented as a fraction a/b.

Sometimes the denominator *b* is a very large integer. Mathematicians hate to write down a tedious long integer, so they usually use two fractions a_1/b_1 and a_2/b_2 to approximate the rational number a/b such that $a_1/b_1 \le a/b \le a_2/b_2$ and the denominators b_1 and b_2 are not greater than a given integer *N*. You are to help mathematicians to find such a_1/b_1 and a_2/b_2 so that the difference between a_2/b_2 and a_1/b_1 is as small as possible.

Input

The input consists of multiple test cases.
Each test case contains exactly one line, which gives three integers *a*, *b* and *N*.
($0 \le a < 2\times10^9, a < b \le 2\times10^9,\ 1 \le N < 2\times10^9$).
The last test case is followed by a line containing three zeros.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by the two fractions a_1/b_1 and a_2/b_2 formatted as the sample output.

The greatest common divisor of a_1 and b_1 must be 1, so does that of a_2 and b_2.

Sample Input

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

Sample Output

```
Case 1: 0/1 1/1
Case 2: 0/1 1/2
```

Problem H: Triangles

There are some points in the 2-dimensional plane. Each point is colored red, green or blue. No three points are collinear. Any triple of blue points can form a triangle, which is called a blue triangle. A blue triangle is called red-blue triangle if there are more red points than green points in it, or green-blue triangle if there are more green points than red points in it.
You are to count the numbers of red-blue triangles and green-blue triangles.

Input

The input consists of multiple test cases. Each test case starts with a line containing three integers *R*, *G* and *B* ($0 \le R, G, B \le 100$), which are the numbers of red, green and blue points, respectively.
Each of the following *R* lines contains two integers *x* and *y* ($0 \le x, y \le 10000$), which gives the coordinate of a red points.
The following *G* lines and *B* lines give the coordinates of all green points and blue points in the same manner, respectively.
The last test case is followed by a line containing three -1.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by two integers, which are the numbers of red-blue triangles and green-blue triangles.

Sample Input

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

Sample Output

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