2016 Denison Spring Programming Contest
Granville, Ohio
27 February, 2016

Rules:

1. There are **six** problems to be completed in **four hours**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. No whitespace should appear in the output except between printed fields.
4. All whitespace, either in input or output, will consist of exactly one blank character.
5. The allowed programming languages are C, C++, Python 2, Python 3, and Java.
6. All programs will be re-compiled prior to testing with the judges' data.
7. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
8. The input to all problems will consist of multiple test cases.
9. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
10. All communication with the judges will be handled by the PC^2 environment.
11. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A: Dan's Presents

Dan likes to give presents within nested boxes. Say that his present is in a box of size $x \times y \times z$ (where z is the height of the box) and his biggest box is of size $a \times b \times c$ (where c is the height of the box). When he nests a box inside another, the tops are both oriented up, but the inner box may be oriented in either edge-parallel way (see figure below) so that the sides of the inner box are strictly smaller than the sides of the outer one. So, a $2 \times 3 \times 4$ box will fit in a $4 \times 3 \times 5$ box, but not in a $3 \times 3 \times 5$ box or a $3 \times 4 \times 4$ box.

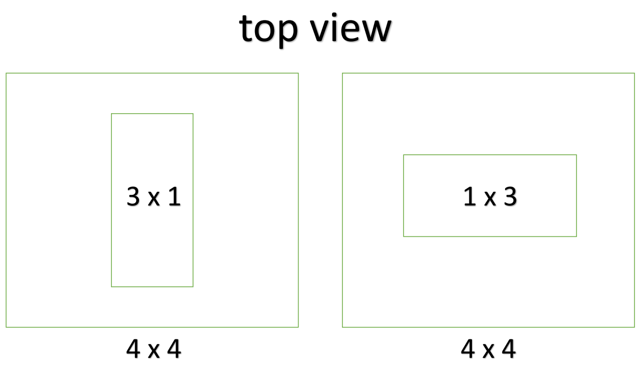


Figure 1: A box may fit inside another in either orientation as long as all dimensions are strictly smaller

Dan wants to nest precisely d boxes between the present and the biggest box. How many ways can he do this, assuming that all boxes have positive integer sides? Note that the top must be oriented up, but we ignore the length/width orientation of boxes nested within each other; that is, we count a $1 \times 2 \times 3$ box fitting inside a $3 \times 4 \times 4$ in only one way, even though it can be placed in either orientation.

For example, if Dan wants to nest one box in between a $2 \times 3 \times 4$ present and a $5 \times 6 \times 6$ box, he can do it in four ways: with a $3 \times 4 \times 5$, a $3 \times 5 \times 5$, a $4 \times 4 \times 5$, or a $4 \times 5 \times 5$ box.

If Dan wants to nest two boxes between a $2 \times 3 \times 4$ present and a $5 \times 7 \times 7$ box, he can do it in three ways:

1. using a $3 \times 4 \times 5$ and a $4 \times 5 \times 6$ box,
2. using a $3 \times 4 \times 5$ and a $4 \times 6 \times 6$ box, or
3. using a $3 \times 5 \times 5$ and a $4 \times 6 \times 6$ box.

Input

Each input will be on a single line with seven integers separated by spaces: $x\ y\ z\ a\ b\ c\ d$, where the present has dimensions $x \times y \times z$, the biggest box has dimension $a \times b \times c$, and d is the number of boxes that will be in between ($1 \leq x, y, z, a, b, c \leq 20$, $1 \leq d \leq 10$). The input cases will be terminated by a single line with seven 0s.

Output

For each case, you should output the case number followed by the number of ways that the boxes can be nested in the manner described above. All cases will be such that the answer does not exceed two billion.

Sample Input

```
2 3 4 3 3 5 1
2 3 4 4 5 6 1
2 3 4 5 6 6 1
2 3 4 5 7 7 2
1 1 1 20 20 20 3
0 0 0 0 0 0 0
```

Sample Output

```
Case 1: 0
Case 2: 1
Case 3: 4
Case 4: 3
Case 5: 161303616
```

Problem B: D-closed Sets

A finite set of positive integers S is division-closed (D-closed) if for all $a, b \in S$, a/b (integer division) and $a \bmod b$ (division remainder) are in $S \cup \{0, 1\}$. For example, it is easy to verify that $\{2, 3, 4, 5\}$ is D-closed, as is $\{2, 3, 4, 5, 7, 9, 14, 29\}$. However, $\{3, 4, 5\}$ is not D-closed since $5 \bmod 3 = 2$ is not in the set. Given a number of integers, you will find the size of the smallest D-closed set that contains them.

Input

Each input will be on a single line, with n (the number of integers, $n > 1$) followed by n integers ($1 < a_1 < a_2 < a_3 < \ldots < a_n \leq 100$) on a single line separated by spaces. The input will be terminated by a single line containing a 0.

Output

For each case, you should output the case number followed by the size of the smallest D-closed set that contains the input numbers.

Sample Input

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

Sample Output

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

Problem C: Jumper

For an arbitrary finite sequence of integers, we can play the following game:

- We start at the first integer and move that many positions to the right in the sequence. If the integer is negative, then we move that many positions to the left.
- Repeat the above process at whatever integer we land on and continue the same process.
- The process will end when you fall off the sequence on the left or right, or it could continue forever.

Your task is to determine which of the three possibilities happens for each input case. For example, for the sequence

2 -2 -1

we will go from 2 to -1 to -2 and then fall off the left of the sequence. For

1 2 3 -1

we go from 1 to 2 to -1 to 3 and then fall off the right of the sequence. Finally, for

3 2 -1 -1

we go from 3 to the second -1 to the first -1 to 2 and then we loop back to the second -1, thus running forever.

Input

Each input will be on a single line. The first integer n ($1 \leq n \leq 1000000$) will be the length of the sequence. The following n integers ($a_1\ a_2\ \ldots\ a_n$, where each $-n < a_i < n$), will be the sequence itself. The input will be terminated by a line with a single 0.

Output

For each case, you should output the case number followed by whether you fall off the "Left" or "Right" or "Neither" (i.e., you loop forever).

Sample Input

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

Sample Output

```
Case 1: Left
Case 2: Right
Case 3: Neither
```

Problem D: Lethal

The goal of the game *Hearthstone*$^{\text{TM}}$ is to use your minions and spell cards to reduce your opponent hero below one health. Even professional Hearthstone players sometimes forget to check for "lethal"—the ability to finish off your opponent—so you will write a program to check this for them.

On a turn, you inflict damage on your opponent hero with your minions and any spell cards you play. You will be given a list of all your minions that are able to damage the opponent hero. The minions will not cost anything to play. You will also have a list of spell cards that are available for you to use. Each spell has a crystal cost as well as a damage value. Your task is to determine whether you have enough damage between your minions and spell cards to win the game.

For example, if your opponent hero has 10 health and you have 3 minions (with 1, 1, and 2 damage) as well as two spells that each cost 2 crystals and do 4 damage and you have a 5 crystal limit, then you have lethal since your minion damage $(1 + 1 + 2 = 4)$ plus your spell damage playable $(4 + 4 = 8)$ exceeds your opponent's health. If you had only one 2 crystal, 4 damage spell in your hand, then you would not have lethal. Also, if you had two spells that did 5 damage and cost 5 crystals each, you wouldn't have lethal since you could only play one of the spells.

Input

Each input will consist of three lines. The first line will have the number of minions you have in play, the number of spells in your hand, the number of crystals you have to spend, and the opponent hero's health ($m\ s\ c\ h$, where $1 \leq m \leq 7$, $0 \leq s \leq 10$, $1 \leq c \leq 10$, $1 \leq h \leq 30$). The second line will have m positive integers indicating the attack damage of each of your minions ($d_1\ d_2\ d_3\ \ldots\ d_m$, $1 \leq d_i \leq 10$ for all i). The third line will have pairs of the damage and cost of your spells $D_1\ C_1\ D_2\ C_2\ D_3\ C_3\ \ldots\ D_s\ C_s$, where $1 \leq D_i \leq 10$ for all i, and $0 \leq C_j \leq 10$ for all j. A line with 4 0s will indicate end of all inputs.

Output

For each input, you should output the case number followed by the string "Lethal" or "No lethal" depending on whether it is possible to defeat your opponent on this turn.

Sample Input

```
3 2 5 10
1 1 2
4 2 4 2
3 1 5 10
1 1 2
4 2
3 2 5 10
1 1 2
5 5 5 5
0 0 0 0
```

Sample Output

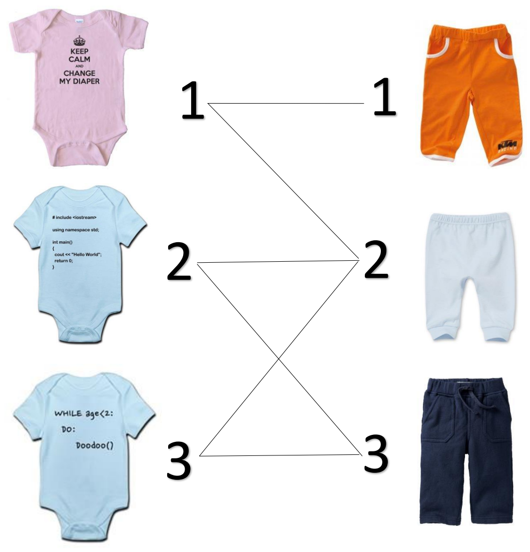
```
Case 1: Lethal
Case 2: No lethal
Case 3: No lethal
```

Problem E: Onesies Twosies

Erin doesn't like doing laundry. She wants to dress her baby in as many matched onesie and pant combinations as possible (wearing each article of clothing at most once) before having to do a wash. The catch is that not every onesie and pant pair match and she doesn't want to take her baby out of the house looking like her dad dressed her.

For this problem, you have to determine the size of the largest collection of pairs of onesies and pants such that each pair is matched and none is used more than once.

For example, if we have that onesie 1 matches pants 1 and 2, onesie 2 matches pants 2 and 3, and onesie 3 matches pants 2 and 3 (see figure below), then Erin can dress her baby in three non-overlapping outfits without doing laundry by matching 1-1, 2-2, 3-3 (or 1-1, 2-3, 3-2).



Input

Each input will consist of multiple lines. The first line will have the number of onesies, the number of pants, and the number of matched pairs (k m n, where $1 \le k \le 100$, $1 \le m \le 100$, $1 \le n \le km$). The following n lines will consist of possible matching onesie/pant pair separated by a single space. The onesies will be numbered $1, \ldots, k$ and the pants will be numbered $1, \ldots, m$. A line with 0 0 0 will terminate all the inputs.

Output

For each input, you should output the case number followed by the number of onesie/pant pairs that can be worn before Erin has to do laundry.

Sample Input

```
3 3 6
1 1
1 2
```

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

Sample Output

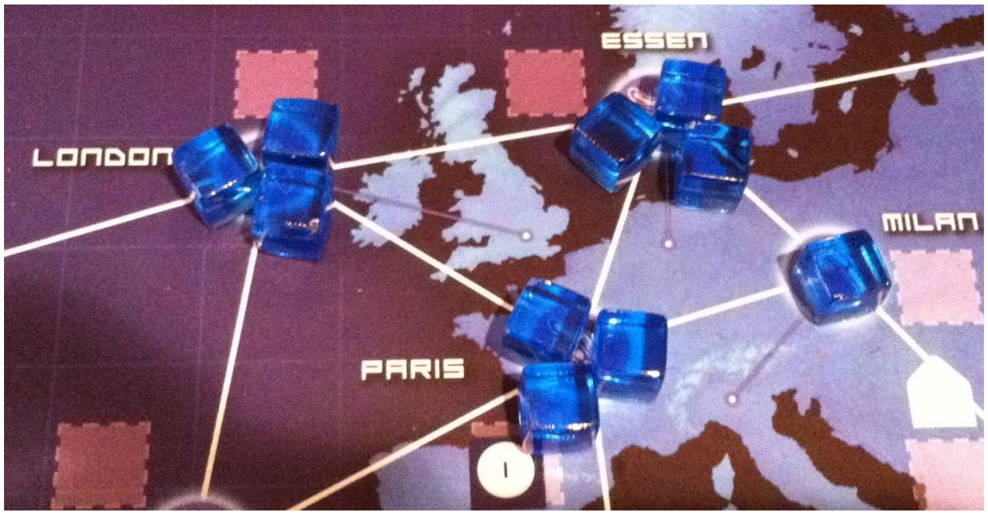
```
Case 1: 3
Case 2: 2
Case 3: 1
```

Problem F: Pandemic

In the board game *Pandemic Legacy*™, you have a number of connected cities in which you are trying to stop the spread of deadly diseases. Any time a city that already has three disease cubes gets infected, it has an outbreak and brings you closer to losing the game. When an outbreak occurs, all the cities connected to the outbreak city get infected, adding another disease cube if it has fewer than three or causing an outbreak if it has three. This can cause outbreaks to cascade across the board in a chain reaction. To protect against outbreaks, players can choose to "quarantine" one city each turn; this prevents that city from having an outbreak. In this problem, you will determine which city should be quarantined.

At the beginning of a turn, each city has between zero and three disease cubes in it. During the turn, the player can put a quarantine on any city on the board. At the end of the turn, a disease cube will be added to a random city, so it is important to place the quarantine strategically.

When an outbreak occurs in a city, the infection is spread to all the cities it is connected with that have not had an outbreak yet this turn. Note that a city can only have one outbreak per turn, preventing indefinite chain reactions. However, it is possible for a city to get multiple disease cubes in a turn. Consider the following example:



Suppose an outbreak hits London. This causes outbreaks in Essen and Paris. The outbreaks in Essen and Paris both cause cubes to be added to Milan (which will now have a total of 3 cubes). Note, however, that the outbreaks in Essen and Paris don't cause outbreaks in London or one another again (since they have all had outbreaks this turn). Hence, there will have been a total of three outbreaks from London getting hit.

If Essen gets hit first, then there will be outbreaks in London and Paris, also for a total of three outbreaks. Similarly, Paris will cause three outbreaks. Milan getting hit doesn't cause any outbreaks since it only has one cube.

Your goal is to determine which city would cause the largest number of outbreaks if it gets hit at the end of the turn. This is the city you'd probably want to quarantine.

Input

Each input will consist of multiple lines. The first line will have two integers n m ($1 \leq n \leq 100$, $1 \leq m \leq n(n-1)/2$) indicating the number of cities and the number of connections between cities. The cities will be numbered from 1 to n. The following n lines will have integers between 0 and 3 inclusive indicating how many disease cubes there are at each city. The next m lines will consist of pairs of adjacent cities given by their numbers separated by a space. Note that the adjacency connections between cities will go both ways, even though it will only be listed in one way (i.e., as an undirected edge). A line with 0 0 will terminate the input cases.

Output

For each input, you should output the case number followed by the number of the city that should be quarantined (breaking ties by selected the lowest numbered city) as well as the number of outbreaks that would occur if it were the one to be initially infected.

Sample Input

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

Sample Output

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