2019 Denison Spring Programming Contest
Granville, Ohio
2 March, 2019

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 contest 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: Defile

In the game *Hearthstone* you sometimes get to destroy all your opponent's minions using a spell called *Defile*. The spell works by doing 1 damage to all minions and if any minions are destroyed by it, it will do another 1 damage to all minions, repeating this until no more minions are destroyed. For example, in the game below, if the player casts the spell then none of the minions will be destroyed since they all have health above 1. (Note: The health of a minion is the number on its lower right; you can ignore the number on the lower left.)

Figure 1: Example board where Defile doesn't destroy any minions

On the other hand, if Defile is cast on the board below, then the 1-health minion (second from the left in the bottom row) is destroyed, next the two 2-health minions on the top row are destroyed, followed by the three 3-health minions (one on the top row and two in the bottom row). Since this destroys all the enemy minions (all the ones in the top row), this would be considered a successful Defile.

Figure 2: Example board where Defile destroys all enemy minions

The goal of this problem is to determine whether casting Defile will destroy all your opponent's minions. (As a side effect some or all of your minions may get destroyed as well.)

Input

Each input starts with integers n m on a single line. (A line with 0 0 indicates the end of input.) The values n and m ($1 \leq n, m \leq 7$) indicate the number of opponent and player minions, respectively. The second line consists of the health of the n enemy minions e_1 e_2 e_3 ... e_n ($1 \leq e_i \leq 10$ for each e_i). The third line consists of the health of the m player minions p_1 p_2 p_3 ... p_m ($1 \leq p_i \leq 10$ for each p_i).

Output

Each input set should output the case number and a response of either YES or NO depending on whether all the enemy minions will be destroyed or not, as shown in the Sample Output.

Sample Input

```
3 4
3 3 2
7 3 3 3
3 4
2 3 2
7 1 3 3
0 0
```

Sample Output

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

Problem B: Focus

In the board game *Gloomhaven*, player and monster figures move on hexagonal tiles. The monster figures move towards player figures according to focus rules, which you will emulate in this problem.

A monster always tries to move adjacent to the player figure that is nearest in terms of the number of moves it has to take to get to the figure. Moves are always along adjacent hexes. If there are multiple players that can be reached with the same number of moves, the monster moves towards the one with a smaller initiative value. For example, in the figure below (Fig. 3), the monster in the top left will move towards Player 1 because this can be accomplished with fewer moves than to Player 3 (two moves rather than 3) and because Player 1 has a smaller initiative value than Player 2 (10 rather than 20). All players will have distinct initiative values.

Figure 3: Example of monster movement rule

Input

Each input set starts with the integers r c ($1 \leq r, c, \leq 20$) on a line indicating the number of rows and columns on the map. A line with 0 0 indicates the end of input. The second line consists of the location of the monster m_r m_c ($1 \leq m_r \leq r, 1 \leq m_c \leq c$). The third line contains the number of players p ($1 \leq p \leq 10$). The following p lines have each of the player coordinates and initiative values p_r p_c p_i ($1 \leq p_r \leq r, 1 \leq p_c \leq c$, $1 \leq p_i \leq 100$). No two figures (monster or player) will have the same position. All players will have distinct initiative values.

All the positions will be given as row number and column number as shown in the example figure (Fig. 4) below with $r = 4, c = 7$. Note that odd numbered rows have no even numbered columns and vice versa; you may assume that all figure coordinates (both monster and player) will be for existing tiles.

Figure 4: Example of the hexagonal coordinate system ($r = 4, c = 7$)

Output

Each input set should output the case number and the player number (starting at 1) that the monster moves to, as shown in the Sample Output.

Sample Input

```
4 7
1 1
3
1 7 10
4 4 20
3 7 5
4 7
1 1
3
4 4 10
3 1 20
4 6 5
0 0
```

Sample Output

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

Problem C: Iota

In the card game *Iota*, players take turns adding cards in lines to a grid to maximize their point total. In a turn, a player adds 1, 2, 3, or 4 cards in a line, connecting to other cards already in play. (See the figures below for example grids.) All cards must be placed using the following guidelines:

- All played cards must connect in a single straight line and at least one of them must connect with the cards already in play. Note that a line may be extended by adding cards to both ends of an existing line.
- In each line (of two or more cards) created by the placement of cards, each of the attributes (color, shape, number) should be all the same or all different. For example, a line may have cards all with the same color, all distinct shapes, and all the same number. Note that since there are only four colors, shapes, and numbers (and no two cards have the exact same combination of them) no line can have a length greater than four.

The score for some cards that are placed is calculated by adding the face values (numbers) of the lines created. If a card played is part of two lines, then its face value is counted twice. Each four card line created doubles the score for the turn (this can happen multiple times). Additionally, if the player plays four cards in the turn this also doubles the score for the turn.

For example, consider the grid and player cards shown below.

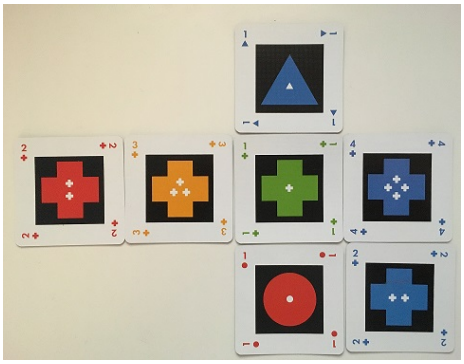

Figure 5: Card grid

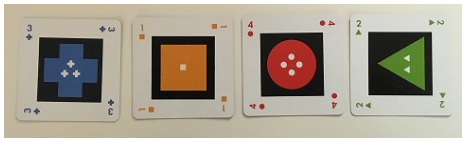

Figure 6: Player's cards

One possible (non-optimal) play would be the one shown below for a total of $7 + 5 = 12$ points (7 points for the two cards played horizontally and another 5 for the vertical line that got created).

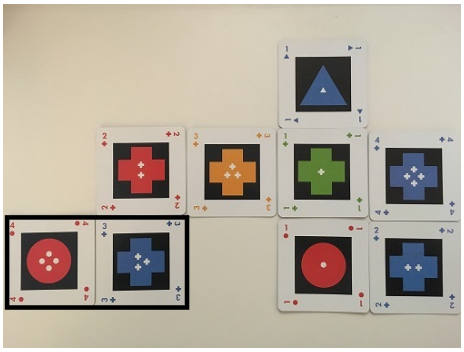

Figure 7: Player turn that results in a score of 12

A slightly better play would be to get a line of 4 (shown below) for a total of 14 (4 points vertically, 3 horizontally, full score doubled for getting a line of 4).

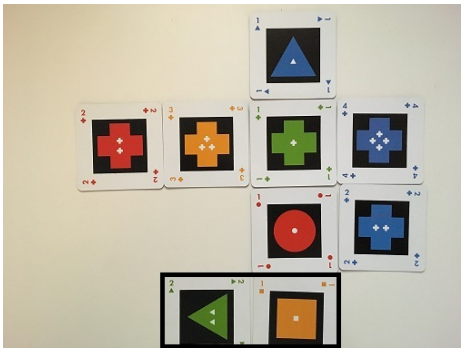

Figure 8: Player turn that results in a score of 14

Of course, it is also possible to use all four of the player's cards as seen in this example with a score of 112 (4 points vertically, 10 points horizontally, full score doubled twice for getting two lines of 4, and then doubled again for using up all four cards).

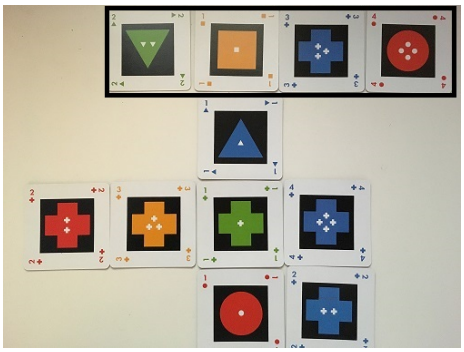

Figure 9: Player turn that results in a score of 112

The optimal play is the following for a score of 184 (two vertical lines worth 4 and 9, 10 points horizontally, full score doubled twice for getting two lines of 4, and then doubled again for using up all four

cards).

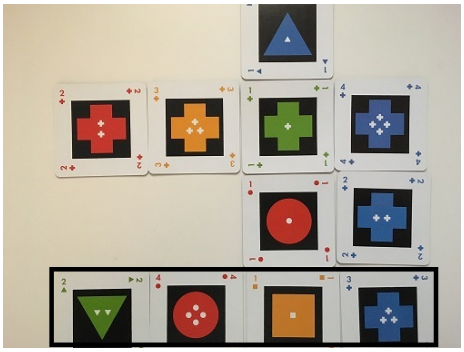

Figure 10: Player turn that results in the top score of 184

For this problem, you will be given an Iota grid in a legal state and a hand of four cards and you have to determine the maximum points that can be earned in that turn.

Input

Each input starts with the number of rows, columns, and pieces on the grid r c p on a single line ($1 \leq r \leq 50, 1 \leq c \leq 50, 1 \leq p \leq 100$). (A line with 0 0 0 indicates the end of input.) The next p lines are the pieces already in the grid given as grid positions followed by the shape, color, and number of the piece r_i c_i s_i k_i n_i ($0 \leq r_i < r, 0 \leq c_i < c, s_i \in \{\text{CIRCLE}, \text{SQUARE}, \text{TRIANGLE}, \text{CROSS}\}, k_i \in \{\text{RED}, \text{GREEN}, \text{BLUE}, \text{YELLOW}\}, 1 \leq n_i \leq 4$). The last four lines of the input are the player's cards, specified by the shape, color, and number format used for the grid.

Output

Each input set should output the case number and the maximum points that can be scored on the grid with the given cards. If no cards can be played legally, then the maximum number of points is zero.

Sample Input

```
3 4 7
0 2 TRIANGLE BLUE 1
1 0 CROSS RED 2
1 1 CROSS YELLOW 3
1 2 CROSS GREEN 1
1 3 CROSS BLUE 4
2 2 CIRCLE RED 1
2 3 CROSS BLUE 2
CROSS BLUE 3
SQUARE YELLOW 1
CIRCLE RED 4
TRIANGLE GREEN 2
1 1 1
0 0 CROSS YELLOW 3
CIRCLE RED 1
TRIANGLE GREEN 2
```

```
CROSS GREEN 1
TRIANGLE BLUE 1
2 2 4
0 0 CROSS BLUE 1
0 1 CIRCLE GREEN 1
1 0 TRIANGLE BLUE 1
1 1 SQUARE YELLOW 1
CIRCLE RED 2
TRIANGLE GREEN 2
CROSS GREEN 2
SQUARE BLUE 2
0 0 0
```

Sample Output

```
Case 1: 184
Case 2: 8
Case 3: 0
```

Problem D: Oral History

The Granville Historical Society has been collecting oral histories for over 200 years. Among the many things they are trying to sort out is when various residents lived in Granville. But oral history being what it is – that is, unreliable – they are worried about inconsistencies. They've decided to start with some very basic checks. Some of the information they get is simply when two people lived in relation to each other. Specifically, given persons A and B, they are told that A lived before B. That is, A died before B was born. Another type of information is that A and B both lived at the same time, meaning only that at some time A and B overlapped; no information regarding when either was born or died. Given a set of information of the two types mentioned, is it consistent? That's the problem you are asked to solve.

For example suppose you are told that A lived before B, B and C overlapped, and A and C overlapped. This would be consistent. (Possible order: A born, C born, A died, B born, C died, B died.) But A lived before B, B and C overlapped, and C lived before A would not be consistent. (The first and last imply that C lived before B, which contradicts the second.)

Input

Each input set starts with the integer n on a line. $n = 0$ indicates the end of input. There follows n lines of information. Each line is of the form $n_1 < n_2$ or $n_1 = n_2$. The former indicates n_1 lived before n_2, the latter that n_1 and n_2 overlapped. Both n_1 and n_2 are uppercase letters, so there is a a limit of 26 possible names. The relationship between any pair of names will appear at most once in any one input case.

Output

Each input set should output the case number and a response of either YES or NO depending if the information is consistent or not, as shown in the Sample Output.

Sample Input

```
3
A < B
B = C
A = C
3
A < B
B = C
C < A
3
A < B
B < C
C < A
0
```

Sample Output

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

Problem E: Rational Moves

Define a "move" on a positive rational number (reduced to lowest terms) to be the addition of 1 to either the numerator or denominator, followed by reduction to lowest terms. For example, from the rational 1/2, you can move to the rational numbers 1/1 and 1/3. For this problem, you will determine whether it is possible to get from one positive rational number to another in a given number of moves.

Input

The input starts with the number of test cases ($n > 0$). The following n lines have the test cases, one per line. Each test case will consist of the positive integers p q r s m ($0 < p, q, r, s \leq 20$, $0 < m \leq 40$) where the starting rational is p/q, the target rational is r/s, and the maximum number of moves is m. You may assume that p/q and r/s are already reduced to lowest terms (i.e., p and q have no common factors and r and s have no common factors).

Output

For each input, you should output the case number and YES or NO depending on whether it is possible to get from p/q to r/s in at most m moves.

Sample Input

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

Sample Output

```
Case 1: YES
Case 2: YES
Case 3: NO
Case 4: YES
```

Problem F: Will Salmon Please Help Me?

As is well known, salmon return to their place of birth to die. For this problem, you will keep track of a population of salmon and determine how long it takes for all of them to die out.

We will suppose that salmon start out on a circular river (that's a thing, right?) as depicted below. Each salmon will be moving either clockwise or counterclockwise at the rate of one cell per minute. Whenever a salmon re-enters its starting cell, it dies and is removed from consideration. When a pair of salmon both occupy the same cell (and neither just died) they both reverse direction.

For example, in the figure below, the river has 12 cells (numbered 0 to 11) and there are two salmon in it, one at position 0 going clockwise and another at position 6 going counterclockwise. After three minutes, the two would both enter cell 3 and in another three minutes both would return to their original cells and die. Thus, the total time for them to die would be 6 minutes.

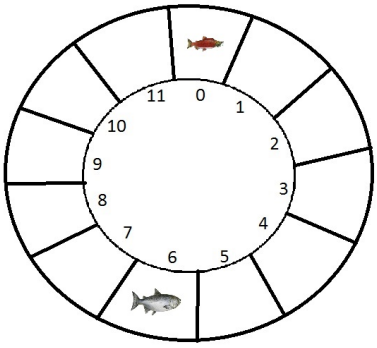

Figure 11: Example 1 for salmon movement

In this second example below, there is a third salmon at cell 11 moving clockwise. Note that this salmon will never occupy the same cell as either of the other two (in the fourth minute step it will enter cell 3 which was just vacated by the other two) and hence it will just take 12 minutes to get back to its starting cell. As a result, the total time for this population will be 12 minutes.

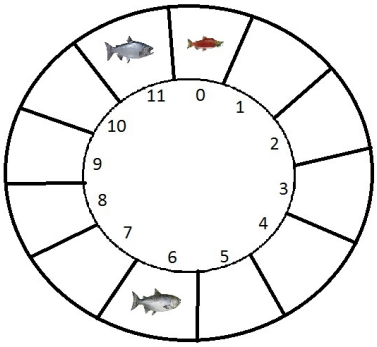

Figure 12: Example 2 for salmon movement

Input

Each input set starts with the length of the river and number of salmon r s ($1 \leq r \leq 1000000, 1 \leq s \leq 100000$). A line with 0 0 indicates the end of input. The next s lines will consist of the starting position

and direction of each salmon. Each starting position will be an integer between 0 and $r - 1$ (inclusive) and the direction will be either a C (clockwise) or CC (counterclockwise). No two salmon will ever start in the same cell.

Output

Each input set should output the case number and the number of minutes for all the salmon to die off. The answer will never exceed 1 billion.

Sample Input

```
12 2
0 C
6 CC
12 3
0 C
6 CC
11 C
0 0
```

Sample Output

```
Case 1: 6
Case 2: 12
```

Extra challenge for after the contest: In all the input cases the salmon will all eventually die out. Is it possible to construct a case in which the salmon bounce around forever? Why or why not? I don't know the answer to this as yet, so please let me know if you find an example where they bounce forever or a proof that they cannot.