2014 Denison Spring Programming Contest
Granville, Ohio
22 February, 2014

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++ 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: Cauldron Cakes

Fred and George are twins who always fight over getting the same number of cauldron cakes. Their mom, Molly, bakes a batch of cauldron cakes in a variety of shapes and sizes, though all cakes somehow have integer weights. To avoid fights between the twins, Molly wants to divide the cakes into two piles so that the total weight of cakes in each pile is the same and the number of cakes in both piles is the same. Given a number of cakes, you are to determine if it is possible to split them into two piles of equal weight and equal number.

For example, if Molly bakes cakes with weights $4, 3, 5, 1, 4, 7, 2, 4$ then there is a solution possible: one pile with $4, 5, 4, 2$ and the other with $3, 1, 7, 4$ (both have a total of 15). On the other hand, there is no way to partition $4, 5, 8, 6$ in a satisfactory fashion.

Input

Each problem instance is given on one line of input and begins with an integer n ($2 \leq n \leq 50$) indicating the number of cakes that Molly baked. This is followed by n integer values (each between 1 and 20) indicating the weights for the n cakes. An input of $n = 0$ indicates the end of problem instances.

Output

For each problem, you are to print the case number and either "YES" or "NO" (one line of output per problem instance) indicating whether there is a feasible solution.

Sample Input

```
8 4 3 5 1 4 7 2 4
4 4 5 8 6
8 3 4 3 4 3 4 3 4
5 3 4 1 2 6
0
```

Sample Output

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

Problem B: Frog and Toad Play a Game

"Toad", says Frog. "I want to play a game".

"Excellent!" says Toad. "How does the game go?"

Figure 1: Frog and Toad Play a Game

Frog explains, "We play on a row of squares numbered 0 to infinity. You start on Square 0, the first one, and I start on Square 1, the second one." Frog goes on, "Then we draw a card from a deck that contains pictures of toads and frogs. If the card is a 'frog', then I get to move. Otherwise if the card is a 'toad', you get to move. We always put the card back in the deck after each draw."

"I like this game, already", says Toad. "How do we move?"

Frog explains, "When we move, we go forward one square. Except if that square is already taken by the other player, then we get to jump over them to the next free square; thus we get to move two squares in this particular case." "The winner", says Frog, "is the player who is furthest along the row after we are done drawing cards."

"Let's play!", exlaims Toad.

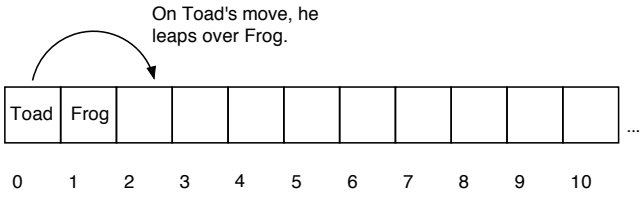

Figure 2: Leap Frog Game

Input

Each problem instance, which is one playing of the game, is given on one line of input composed of an integer n indicating the total number of moves. This is followed by a string of n characters containing only T's and F's. The characters in the string indicate the order in which the cards were drawn for this particular game. An input of $n = 0$ indicates the end of problem instances. You may assume $n \leq 100$.

Output

For each problem, you are to print the case number and either "Toad" or "Frog" (one line of output per problem instance) indicating who won that contest.

Sample Input

```
3 TTF
10 FFTTTFFTFT
0
```

Sample Output

```
Case 1: Toad
Case 2: Frog
```

Problem C: Russian Nesting Dolls

Russian Nesting Dolls (aka Matryoshka Dolls) are a cultural craft that consists of a series of wooden carved dolls. Each doll is hollow inside and can be opened near the middle. The dolls are arranged in size so that each doll can fit inside the one that is next larger in line. All the dolls can be stacked, or nested, inside the largest doll.

Figure 3: Russian Nesting Dolls

In this problem, you desire to count how many ways you can nest the dolls to create a specific number of stacks. For example, assume you have four dolls and you wish to create two stacks. Assuming the dolls are numbered smallest to largest as 1 to 4, one way you can create two stacks is to have the largest doll be one stack, and the three smallest dolls be the second stack (the two smallest dolls are inside the second largest). We represent this configuration as (4),(3-2-1). Or you can nest the two larger dolls in one stack and the two smaller dolls in the other stack: (4-3), (2-1). In fact, there are seven unique ways to nest four dolls into exactly two stacks:

(4),(3-2-1) (4-3),(2-1) (4-2),(3-1) (4-1),(3-2) (4-3-2),(1) (4-2-1),(3) (4-3-1),(2)

Input

Each instance of input begins with an integer n indicating the number of dolls. This is followed by integer m indicating the number of stacks. An value of $n = 0$ indicates the end of input. (Note that there is no m value when $n = 0$.) You may assume that $1 \leq n \leq 12$ and that $1 \leq m \leq n$.

Output

For each problem instance you are to print the case number and then the exact number of unique ways to stack n dolls into exactly m stacks.

Sample Input

```
4 2
4 3
0
```

Sample Output

```
Case 1: 7
Case 2: 6
```

Problem D: Tim's Toy Trains

Tim has a number of toy trains that he needs help powering. Each of his n trains uses exactly k batteries. The power output of each train is the minimum power of its k batteries. Given a number of batteries with known power, Tim wants to find an assignment that minimizes the difference in power output between the highest and lowest powered trains.

For example, if $n = 3, k = 2$, and Tim has batteries with power $3, 5, 1, 7, 16, 5, 4, 3, 8$, he can assign batteries with power 3 and 7 to the first train (resulting in a power of 3), then 3 and 8 to the second train (resulting in a power of 3), and 4 and 16 to the last train (resulting in a power of 4). The difference between the highest power train and the lowest power train is $4 - 3 = 1$.

Input

The input for each case will consist of two lines. The first line will have the values of n and k (both positive integers, $2 \leq n \leq 100$, $1 \leq k \leq 100$). The second line will consist of the number of available batteries t ($nk \leq t \leq 10000$), followed by the power of these t batteries, each a positive integer of at most 100. The line 0 0 will follow the last test case.

Output

For each test case, generate one line of output, in the format given below, giving the minimum difference between the maximum and minimum powered trains when assigned optimally.

Sample Input

```
3 2
9 3 5 1 7 16 5 4 3 8
2 2
5 1 4 4 4 4
3 1
3 100 2 1
0 0
```

Sample Output

```
Case 1: 1
Case 2: 0
Case 3: 99
```

Problem E: Virtual Realty

You and your friends have developed a realty startup that will make use of your computer science skills. You have listings for n houses, each with a square footage and asking price, and m clients looking to buy a house, each with a minimum square footage requirement. Your job is to assign houses to clients in such a way that the total price of all the houses sold is maximized. Note that not every house has to be sold and not every client has to get a house.

For example, if the house and client attributes are as in the tables below, then the optimal solution is to sell House 3 to Client 2 and House 4 to Client 3 for a total sale price of $200000 + 300000 = 500000$ dollars.

House	Sq. Footage	Price (dollars)
1	1500	150000
2	1700	250000
3	1800	200000
4	1700	300000

Client	Min. Sq. Footage
1	1900
2	1800
3	1600

Figure 4: Example of the realty problem

Input

Each instance is on two lines. The first line will have the number of houses, n, followed by the n pairs of square footage and prices: $n\ a_1\ p_1\ a_2\ p_2\ \ldots\ a_n\ p_n$. All values are integers with $1 \leq n \leq 100$ and all $500 \leq a_i \leq 5000$, $10000 \leq p_i \leq 1000000$. The second line will have the number of clients, m, followed by the m minimum square footage values for each client: $m\ b_1\ b_2\ \ldots\ b_m$. All these values are integers with $1 \leq m \leq 100$ and all $500 \leq b_i \leq 5000$. The input will be terminated by a line with a single 0.

Output

For each problem instance you are to print the case number and the sum of the prices of all the houses that were successfully matched in an optimal solution.

Sample Input

```
4 1500 150000 1700 250000 1800 200000 1700 300000
3 1900 1800 1600
0
```

Sample Output

```
Case 1: 500000
```

Problem F: Wall Heights

Alice and Bob are two kids that grew up in their house on 100 W College Street. As the kids were growing taller, their parents marked their heights by having them stand against the kitchen wall. Each height is a line that represents one of the two kids' heights at a specific time. Each line is identified by an "A" or "B" indicating who that line is for, but unfortunately there is no date with each line. The parents do know that they followed two rules: (1) each kid has the same number of marks, and (2) each marking was done in a pair – that is, Alice and Bob each received a mark on the same date. You can also assume that each kid kept growing (no one shrank during this period, so the marks for Alice and Bob are monotonically increasing respectively). Your job is to reconstruct the history of the markings to indicate which child was taller as they were growing up.

Input

Each problem instance is given on one line of input and begins with an integer k indicating k *pairs* of markings ($1 \leq k \leq 1000$). That is followed by a string a length $2k$ which contains exactly k A's and k B's. The string represents the markings on the wall as read from lowest to highest. A value of $k = 0$ indicates the end of input. Notice that no two markings will be the same height (no ties).

Output

For each problem instance you are to print the case number and then choose one of three possibilities:

- If the input indicates that Alice was always taller than Bob, then print "Alice".
- If the input indicates that Bob was always taller than Alice, then print "Bob".
- If the input indicates that Alice was sometimes taller than Bob and also that Bob was sometimes taller than Alice, then print "Mixed".

It will always be possible to determine which of the above three possibilities is true for the input.

Sample Input

```
2 BBAA
2 ABBA
7 BBABAABABABBAA
0
```

Sample Output

```
Case 1: Alice
Case 2: Mixed
Case 3: Alice
```