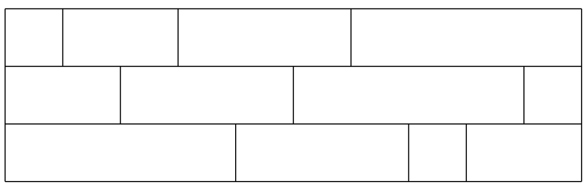
2013 Denison Spring Programming Contest
Granville, Ohio
23 February, 2013

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: Brick Laying

Suppose you have bricks of lengths 1, 2, 3, and 4. Consider the wall of width 10 and 3 rows tall illustrated below. Note one brick of each length is used in each row and, curiously, no two vertical seams align on any two rows. We'll call such a wall a *legal wall.*



Doesn't this makes you wonder for what other lengths of bricks is this possible? I thought so. It turns out that this is a difficult problem to solve in general and there are some interesting open questions concerning it. We'll try something a little more tractable. Here, you'll be given a legal wall made up of one or more rows of n distinct lengths of bricks and wish to know if it is possible to add one more row to the wall, using those n bricks, so the new wall is still legal. (There may be more than one way to do this.)

Input

Input for each test case will be on one line of the form $n\ r\ \ell_1\ \ell_2\ \ldots\ \ell_{rn}$, where $1 < n, r \leq 8$ and $\ell_i < 100$. The lengths ℓ_i will be arranged so the first n consist of the bricks in the first row, the next n the bricks in the 2nd row, and so forth for all r rows. A line with 0 will follow the last test case.

Output

Output one line for each test case in the format shown below indicating if it possible to add a row to the wall.

Sample Input

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

Sample Output

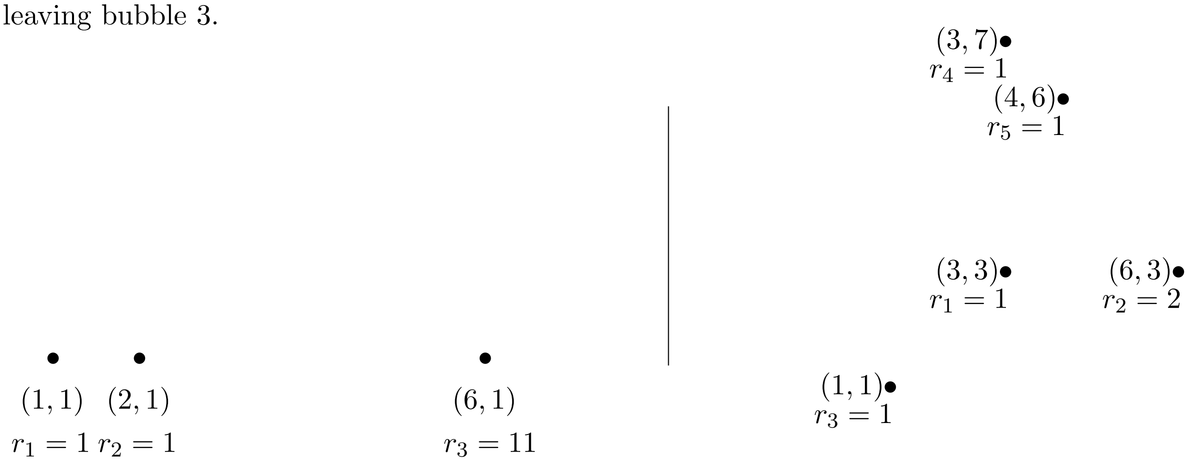
```
Case 1: YES
Case 2: NO
Case 3: NO
```

Problem B: Bubbles

A soap bubble starts as a point and expands equally in all directions until it hits another soap bubble, at which time both of them burst and disappear. Each bubble grows at a constant rate, but the rates for different bubbles may be different depending on the type of soap used.

Suppose an odd number of soap bubbles are placed at distinct points on a plane. Your task is to determine, after all other pairs of bubbles have burst, which is the last bubble remaining. You may assume that any bubble will meet at most one other bubble at any given time. (So, one bubble will not burst two or more other bubbles.)

Consider the two examples below, with initial locations and rate of expansions given. In the figure on the left, the second and third bubbles will hit each other after 1/3 second, and only the the first bubble will remain. For the figure on the right, note that bubbles 4 and 5 touch first, then bubbles 1 and 2, leaving bubble 3.



Input

Input for each test case will be on one line of the form $n\ x_1\ y_1\ r_1\ x_2\ y_2\ r_2\ \ldots\ x_n\ y_n\ r_n$, where (x_i, y_i) is the coordinate of the ith bubble and r_i is the rate at which it grows (in unit distance per second). Each x_i, y_i, r_i is a positive integer no greater than 1000 and n is an odd integer with $0 < n < 1000$. A line with 0 will follow the last test case.

Output

Output one line for each test case in the format shown below indicating the bubble that survives longest.

Sample Input

```
3 1 1 1 2 1 1 6 1 11
5 3 3 1 6 3 2 1 1 1 3 7 1 4 6 1
0
```

Sample Output

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

Problem C: Football

A team can score in many ways in American football. One can score 3 points for a field goal (FG), 2 points for a safety (S), and 6 points for a touchdown (TD). When scoring a TD, the scoring team may attempt to kick an extra point (XP) for 1 point, or a 2-point conversion (2PC), for a surprising 2 points. In addition, in the NFL or college (but not in high school) the defense can score if they either block an XP attempt or recover a fumble or intercept a pass on a 2PC attempt and run the ball back to their own end zone. (We'll call this a run back (RB).) In this case they get 2 points.

Suppose you see that team A beats team B by a score of 8-5. How many ways could this have happened? There are 5: TD+2PC/FG+S, FG+FG+S/FG+S, TD+S/FG+S, S+S+S+S/FG+S, or TD+S/FG+RB. You can check that there are also 5 ways to get a 9-5 score.

In this problem, you are simply going to calculate the number of different ways a given football score could occur.

Input

Input for each test case will be one line of the form A B, indicating that team A scored A points and team B scored B points. Neither score will exceed 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 number of ways the given score could occur.

Sample Input

```
8 5
5 9
0 6
1 0
0 0
```

Sample Output

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

Problem D: Goldbach Spread

You may have heard of the Goldbach conjecture: Every even integer larger than 2 is the sum of two primes. For example, $6 = 3 + 3$, $8 = 5 + 3$ and $30 = 13 + 17$. While this conjecture has been confirmed for large even integers, it remains unproven. For all but the smallest evens there is usually many ways to express the integer as the sum of two primes. For example, $100 = 3+97 = 11+89 = 17+83 = \cdots = 47+53$. Note that the closest the two primes are here is 6 ($= 53-47$) while the farthest is 94 ($= 97-3$). The difference between these two values we'll call the *Goldbach Spread*. Thus the Goldbach Spread of 100 is 88 and the Goldbach Spread of 8 is 0, since 8 is uniquely the sum of two primes. Your job here is to find the Goldbach Spread of various even integers.

Input

Input for each test case will be an even integer larger than 2 and no more than 1,000,000 on one line. A line with 0 will follow the last input.

Output

For each test case output the Goldbach Spread of the integer using the format given below.

Sample Input

```
100
8
30
0
```

Sample Output

```
Case 1: 88
Case 2: 0
Case 3: 12
```

Problem E: Skippy

Skippy is a two-player game whose "board" is a long string of characters, which is traversed via a game token left-to-right, until coming to the end. After each move in the game, the token is "sitting" at a particular place in the string, which we'll call the current position (CP). (Initially the token is sitting to the left of the leftmost character.) In addition to the string, the players are given a positive integer k. On a player's turn, she picks a character c and then moves to the kth occurrence of character c to the right of the CP.

Player A, moving first, tries to end the game as quickly as possible and so picks the character that will move the token furthest to the right. Player B, on the other hand, wants the game to go as long as possible and so picks the character that would move the token the smallest distance to the right. The game ends when a move either places the token at the rightmost character of the string, or the move moves the token past the end of the string. (We could think of this as the game ending in the middle of a turn, which often happens.) You are going to determine the number of turns in this particular game.

For example, let's suppose the string is `abbabbbaaabbba` and $k = 2$. Player A will pick the character `a`, which moves the CP to position 4. Player B then picks `b`, moving CP to position 6. A then picks `b`, moving CP to position 11. B again picks `b`, moving CP to position 13. Finally A picks either character which moves CP past the end of the string, ending the game in 5 turns.

Input

Input for each test case is given on one line of the form $a\ n\ k\ s$ where a, n, and k are positive integers and s is a string. s is a string of length n over the alphabet consisting of the first a lower case letters. $1 \leq a \leq 26$, $1 \leq n \leq 5000$, and $1 \leq k \leq 100$. A line with 0 will follow the input for the last test case.

Output

Output one line for each test using the format given below, indicating the number of turns taken by the players using the strategies described above.

Sample Input

```
2 14 2 abbabbbaaabbba
2 5 2 abbab
3 10 1 abbabcacba
4 15 3 abdcbcbabcdbdac
0
```

Sample Output

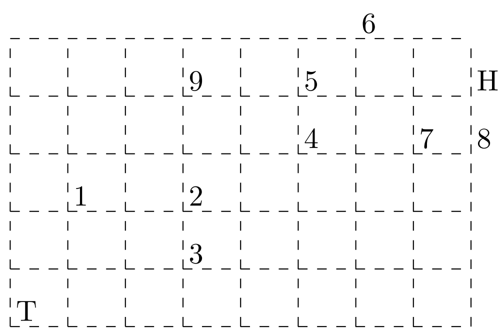
```
Case 1: 5
Case 2: 2
Case 3: 3
Case 4: 2
```

Problem F: Trip to Squaresville

Sam and Sara are visiting Squaresville from their home in Smalltown. They arrive on Saturday morning and are only there for the weekend, so wish to see as many sights as possible. Sam and Sara have a suite at the Squaresville Sheraton, some blocks away from the train station. They've decided to walk to the hotel and, while traveling the shortest distance possible, they want to see as many sights as they can. Given their list of "must see" sights, you're going to determine how many they can see on the way to their hotel.

The streets of Squaresville are on a perfect square grid (naturally) with streets running north-south or east-west only, evenly spaced. The sights are all conveniently located at intersections.

For example, consider the possible map of Squaresville given below. `T` marks the train station, `H` marks the hotel and each sight is marked with a number. Note that Sam and Sara could manage to visit as many as 5 sights. For example, they could visit in order 1-2-4-7-8, or alternatively visit 3-2-4-7-8. But they could visit no more than 5 sights unless they were willing to travel farther, which they are not.



Input

Input for each test case will be on one line of the form $n\ x_T\ y_T\ x_H\ y_H\ x_1\ y_1\ \ldots\ x_n\ y_n$, where (x_T, y_T) is the x-y coordinate of the train station, (x_H, y_H) is the coordinate of the hotel and (x_i, y_i) are the coordinates of each of the sights. Each x and y is a positive integer no greater than 1000 and $0 < n \leq 100$. You may assume the train station, hotel, and all the sites are at distinct locations. A line with 0 follows the input for the last test case.

Output

Output one line for each test case, using the format given in the sample below, giving the maximum number of sights Sam and Sara can visit.

Sample Input

```
9 1 1 9 5 2 3 4 3 4 2 6 4 6 5 7 6 8 4 9 4 4 5
6 8 14 3 10 2 11 2 12 5 1 6 1 7 15 9 12
0
```

Sample Output

```
Case 1: 5
Case 2: 0
```