22nd Annual Denison Spring Programming Contest
Granville, Ohio
26 February, 2011

Rules:

1. There are **six** questions 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. The allowed programming languages are C, C++ and Java.
4. All programs will be re-compiled prior to testing with the judges' data.
5. 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).
6. The input to all problems will consist of multiple test cases unless otherwise noted.
7. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
8. All communication with the judges will be handled by the PC^2 environment.
9. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A: Something's Shifty

Emma is decrypting messages. One simple method is to use a simple shift cipher. In a shift cipher, each letter of the alphabet is replaced by another. You determine which by shifting the alphabet up by a certain number of letters s. For example, if $s = 2$, an A would be replaced by a C (2 letters up from A), a B by D and so on until X is replaced by Z. Now for Y and Z, we simply start over at the beginning of the alphabet: Y is replaced by A and Z is replaced by B.

Thus, for a shift of 2, the message `I LIKE CHOCOLATE` is encrypted as `K NKMG EJQEQNCVG`. But Emma sees no word breaks; the letters appear as a continuous string: `KNKMGEJQEQNCVG`, giving her no clues as to where the words might be.

Now if the method used is really a shift cipher, finding the shift is not hard, but it is a little tedious. Emma's current method is to simply try them all until she finds one that works. These messages are of modest length – no more than 100 characters long. Emma reasons that the original message has a good chance of having the word `THE` or the word `AND` in it. She wants to find if there is a shift that will result in a message where either of these words or both appear. Your job is to help Emma out here.

Input

Input for each test case will consist of a line of the form n s where n ($n \leq 100$) is the length of the decrypted text and s is the decrypted text given as a string (of upper case letters) of length n. A line containing zero follows the last test case.

Output

You should produce one line of output for each test case, using the format below, indicating if there is indeed a shift where the message contains either `THE`, `AND`, or both. Note that 0 will never be a possible shift (as then the decrypted text and the original are one and the same, which is not very secure). You should print `NO` or `YES` accordingly.

Sample Input

```
14 KNKMGEJQEQNCVG
29 KYZJZJSZXREUSRUKFKIPKFUVTIPGK
41 FRJXXTKRNHJKTZSIFHFHMJTKHMJJXJNSYMJUFSYWD
6 ZMCVJG
0
```

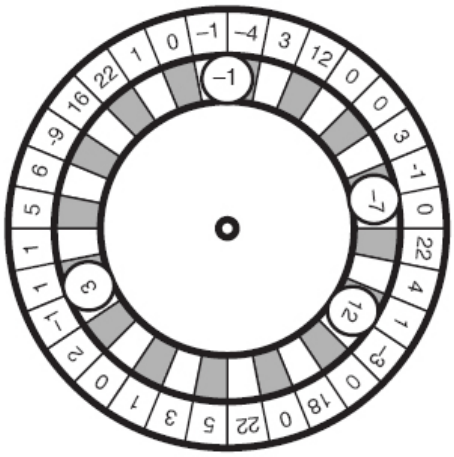
Sample Output

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

Problem B: Wheel Goes Round and Round

A crazy version of roulette, which we'll call Crazy Roulette, is played with a wheel with S slots, each numbered with an integer between -64 and 64, inclusive. Four balls are successively rolled by the dealer into the spinning wheel, each one also has a number from -64 to 64. Four players will play each game, each player betting on one of the balls. So, each ball has a player and each player, a ball.

After spinning, the balls will rest on two adjacent slots of the wheel, as shown in the picture below, which shows a wheel with 32 slots. Balls end up in the relative clockwise position in which they are rolled into the wheel. In this example, the balls were rolled in order 12, 3, -1, and -7. The value of a ball in a turn is calculated by multiplying the ball's number with the sum of the two slots it straddles. So the balls in the example have values -24, 0, 5, and 7. If the value is positive, the dealer takes that money from the player; if negative, the dealer pays that money to the player; if 0, no money changes hands.



You will be asked to find the dealer's maximum profit given the wheel and the sequence the balls are rolled. In our example, a maximum of 558 can be realized if the first ball straddles 16 and 22, the second ball straddles 22 and 4, the third straddles -3 and 0 and the last ball straddles 6 and -9.

Input

Input for each test case will consist of 2 lines. The first line will be of the form $s\ n_1\ n_2\ \ldots\ n_s$ where s ($8 \leq s \leq 100$) is the number of slots on the wheel and $n_1\ n_2\ \ldots\ n_s$ are the values of the slots in clockwise order. The second line gives the value of the four balls in the order rolled. A line containing zero follows input for the last test case.

Output

Produce one output line for each test case, in the format shown below, giving the maximum profit a dealer can have.

Sample Input

```
32 -1 -4 3 12 0 0 3 -12 0 22 4 1 -3 0 18 0 22 5 3 1 0 2 -1 1 1 5 6 -9 16 22 1 0
12 3 -1 -7
0
```

Sample Output

```
Case 1: 558
```

Problem C: Aren't We Special

An Extra Special Number (ESN) is an integer with $2n$ digits that is the product of 2 n-digit numbers whose digits, when combined, are a permutation of the original digits. The smallest ESN is 1260 (21×60). Technical point: We also require that an ESN cannot end in more than one zero. (126000 is not an ESN.) Another ESN is 10052010 (2010×5001).

Input

Input for each test will be a line containing two positive integers a and b ($1000 < a < b < 100000000$). You may assume that $b - a <= 100000$. A line containing 0 0 follows the last input line.

Output

Each test case should produce one line of output, in the format given below. This line should have the number of ESN's e where $a \leq e \leq b$ and, if this count is bigger than zero, the value of the largest ESN in this range.

Sample Input

```
1200 10000
100000 200000
11000000 11100000
10000000 10050000
0 0
```

Sample Output

```
Case 1: 7 6880
Case 2: 54 197725
Case 3: 6 11085304
Case 4: 2 10042510
```

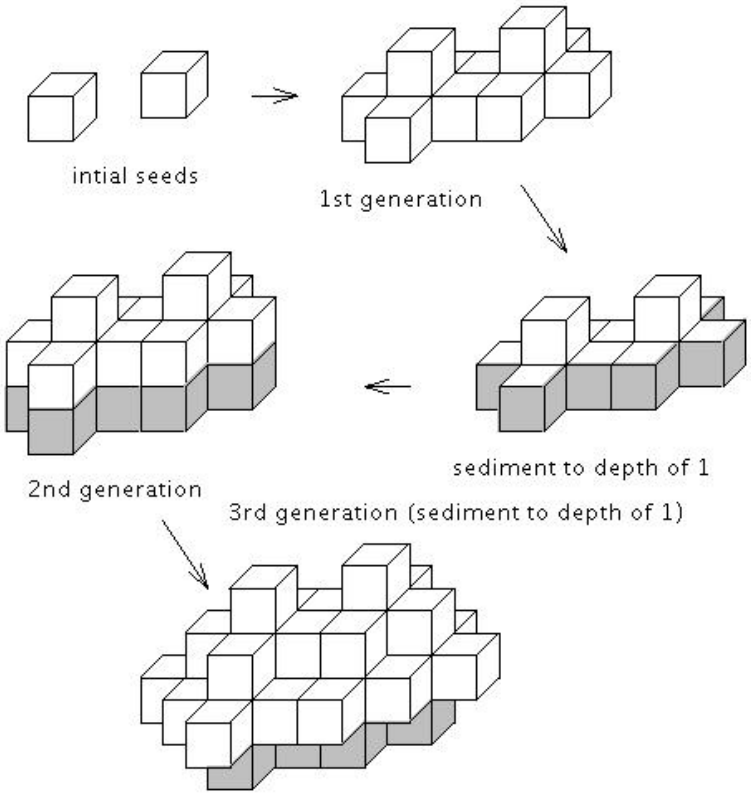
Problem D: Stuff Growing on Pond Bottoms

A now extinct primitive coral-like organism called Stromatoporiods (Stroms) grew on the bottom of ponds and streams. We will model Stroms as collections of cubes; we'll call a single cube a *cell*. Now at each time period, every face of a Strom cell that is exposed to water will grow a new cell. Those faces exposed to mud or against other Strom cells, will not grow. For example, a single cell at the bottom of the pond will have 5 faces exposed to water and so will sprout 5 new cells. Notice the old cell will grow no more since none of its 6 faces are now exposed to water.

Now periodically, sediment falls into the pond, due to runoff from rain. Those faces of Strom cells covered by sediment will not grow after that. Indeed, a Strom cell entirely covered will stop growing and die. (The fossils of Stroms are found under sediment.)

So, imagine the bottom of a pond as a grid. You will be given locations of single-celled Stroms along the bottom. (Think of these as the seeds to the colony.) These will grow at each time increment. Assume that two cells with one open cube between them will create a cell in that intervening location in the next time period; there will be no conflict. You will also be given a series of sediment deposit events which will tell you the amount of sediment deposited at various times. These deposits will be *at the end* of each time period; that is, after the Stroms have grown. Furthermore, if a cell is, say, two removed from the bottom – that is, is sitting on a cell that is sitting on the bottom – sediment of depth 2 will not cover the top face of this cell, only the sides of the cell. Note that the sediment builds up over time; that is, sediment of depth 2 deposited over sediment of depth 3 results in a total sediment depth of 5.

As an example, suppose the colony is seeded at locations $(1, 1)$ and $(3, 2)$. After the 1st time step there is a sediment deposit of depth 1. Then the first three generations are depicted below.



Input

Input for each test case will be on three lines. The first line will consist of three positive integers, S E T, indicating there are S seeds, E sediment events and the simulation will run for T time steps. ($S < 20$, $E < 10$, $T < 200$.) The 2nd line will be of the form x_1 y_1 x_2 y_2 $\ldots$ x_S y_S, where (x_i, y_i) is the coordinates of the ith cell. All values are integers and $-100 < x_i, y_i < 100$. The 3rd line is of the form t_1 d_1 $\ldots$ t_E d_E indicating that at the end of time period t_i sediment of depth d_i is deposited. All are positive integers, and $t_1 < t_2 < \cdots < t_E < T$. Total sediment depth will never exceed 200. A line with three zeros follows the last test case.

Output

For each test case, output one line containing the total number of cells, both living and dead, in the colony, using the format given below.

Sample Input

```
2 1 3
1 1 3 2
1 1
1 3 4
1 1
1 2 2 1 3 1
5 3 100
-2 -2 -1 -1 0 0 1 1 1 10
3 2 15 1 27 13
0 0 0
```

Sample Output

```
Case 1: 44
Case 2: 9
Case 3: 508087
```

Problem E: Creases

If you have a strip of paper (of unit length, say) you can fold it in half and put a crease on the fold so the crease will be exactly 1/2 from each end. Now suppose we start with a crease 1/5 from one end (say the left end), dividing the strip of paper into parts of length 1/5 and 4/5. Now the 4/5 length can be halved (by folding the right side over to the last crease we made), putting a crease 3/5 from the left. Now this leaves 2/5 on the right part, which we can halve, putting a crease 4/5 from the left. We continue in this manner, always folding the part over with the even numerator to the newly-created crease. Eventually, we'll get back to the original crease 1/5 from the left. In the case of starting at 1/5, we'd make creases at distances, 3/5, 4/5, 2/5, and 1/5 from the left, in that order.

As another example, let's start with a crease 1/7 from the left. (The denominator of course must be odd, here.) In this case, we'd make creases at distances 4/7, 2/7, and 1/7.

The problem here is given n (n odd), find the creases you'd make when following this process. You'll print how many creases you make before returning to $1/n$. (You will always return to $1/n$.)

Input

Each test case will consist of a single odd integer less than 1,000,000 and greater than 2. A line with 0 will follow the last test case.

Output

The output for each test case will be in the format given below, giving the number of creases you make before making that final crease at $1/n$. Notice that you count neither the first nor last crease you make at distance $1/n$.

Sample Input

```
5
7
1001
999999
979897
0
```

Sample Output

```
Case 1: 3
Case 2: 2
Case 3: 59
Case 4: 179
Case 5: 57639
```

Problem F: Bridges over Untroubled Waters

The countries of Umbrio and Yumbrio are separated by the great river Bumbrio, which runs west-east. Near the Bumbrio, on the Umbrio side, there are n cities. On the Yumbrio side there are also n cities. In the spirit of fostering trans-brio relations, each city on the Umbrio side is paired with a sister city on the Yumbrio side. Now, the two countries want to build bridges across the Bumbrio between sister cities. The restriction is no two bridges should cross.

For example, suppose the Umbrio cities are A, B, C, D, E, F, and G (west-to-east) and the Yumbrio cities are f, e, a, g, d, c, and b (west-to-east) where the sister cities have the same letters. Note that we could build bridges from A to a, and C to c, but no more could be built. Indeed, no more than two bridges can be built no matter how you try. You are to determine the maximum number of bridges that can be built given various sister city alignments.

Input

Input for each test case is on one line. The first number on the line is a positive integer n ($n < 1000$), followed by a permutation of $\{1, 2, \ldots, n\}$. This permutation indicates the west-to-east order of the n cities on the Yumbrio side labeled as to their sister city on the Umbrio side where the Umbrio cities are in west-to-east order $1, 2, \ldots, n$. A line with zero follows the last test case.

Output

Each test case should produce one line of output, in the format given below, giving the maximum number of bridges that can be built.

Sample Input

```
7 6 5 1 7 4 3 2
10 10 9 8 7 6 5 4 3 2 1
5 1 2 5 3 4
0
```

Sample Output

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