1.Find maximum value of Sum( i\*arr[i]) with only rotations on given array allowed

Given an array, only rotation operation is allowed on array. We can rotate the array as many times as we want. Return the maximum possbile of summation of i\*arr[i].

Examples :

Input: arr[] = {1, 20, 2, 10}

Output: 72

We can 72 by rotating array twice.

{2, 10, 1, 20}

20\*3 + 1\*2 + 10\*1 + 2\*0 = 72

Input: arr[] = {10, 1, 2, 3, 4, 5, 6, 7, 8, 9};

Output: 330

We can 330 by rotating array 9 times.

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

0\*1 + 1\*2 + 2\*3 ... 9\*10 = 330

2.Print the pattern

Input : 4

4 4 4 4 4 4 4

4 3 3 3 3 3 4

4 3 2 2 2 3 4

4 3 2 1 2 3 4

4 3 2 2 2 3 4

4 3 3 3 3 3 4

4 4 4 4 4 4 4

Input: 3

3 3 3 3 3

3 2 2 2 3

3 2 1 2 3

3 2 2 2 3

3 3 3 3 3

3. Given an array of integers, sort the array according to frequency of elements. For example, if the input array is {2, 3, 2, 4, 5, 12, 2, 3, 3, 3, 12}, then modify the array to {3, 3, 3, 3, 2, 2, 2, 12, 12, 4, 5}.

4. Find the smallest window in a string containing all characters of another string

Given two strings string1 and string2, find the smallest substring in string1 containing all characters of string2 efficiently.

For Example:

Input : string = "this is a sample text for testing"

pattern = "exr"

Output : Minimum window is "ext for"

Explanation: "t stri" contains all the characters

of pattern.

Input : string = "geeksforgeeks"

pattern = "ork"

Output : Minimum window is "ksfor"

5. Given a set of time intervals in any order, merge all overlapping intervals into one and output the result which should have only mutually exclusive intervals. Let the intervals be represented as pairs of integers for simplicity.

For example, let the given set of intervals be {{1,3}, {2,4}, {5,7}, {6,8} }. The intervals {1,3} and {2,4} overlap with each other, so they should be merged and become {1, 4}. Similarly {5, 7} and {6, 8} should be merged and become {5, 8}

6. You are the head cook on duty. The pancakes are cooked in a single row over a hot surface. As part of its infinite efforts to maximize efficiency, the House has recently given you an oversized pancake flipper that flips exactly K consecutive pancakes. That is, in that range of K pancakes, it changes every happy-side pancake to a blank-side pancake, and vice versa; it does not change the left-to-right order of those pancakes.

You cannot flip fewer than K pancakes at a time with the flipper, even at the ends of the row (since there are raised borders on both sides of the cooking surface). For example, you can flip the first K pancakes, but not the first K - 1 pancakes.

Your apprentice cook, who is still learning the job, just used the old-fashioned single-pancake flipper to flip some individual pancakes and then ran to the restroom with it, right before the time when customers come to visit the kitchen. You only have the oversized pancake flipper left, and you need to use it quickly to leave all the cooking pancakes happy side up, so that the customers leave feeling happy with their visit.

Given the current state of the pancakes, calculate the minimum number of uses of the oversized pancake flipper needed to leave all pancakes happy side up, or state that there is no way to do it.

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each consists of one line with a string S and an integer K. S represents the row of pancakes: each of its characters is either + (which represents a pancake that is initially happy side up) or - (which represents a pancake that is initially blank side up).

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is either IMPOSSIBLE if there is no way to get all the pancakes happy side up, or an integer representing the the minimum number of times you will need to use the oversized pancake flipper to do it.

Limits

1 ≤ T ≤ 100.

Every character in S is either + or -.

2 ≤ K ≤ length of S.

Small dataset

2 ≤ length of S ≤ 10.

Large dataset

2 ≤ length of S ≤ 1000.

Sample

Input

Output

3

---+-++- 3

+++++ 4

-+-+- 4

Case #1: 3

Case #2: 0

Case #3: IMPOSSIBLE

In Case #1, you can get all the pancakes happy side up by first flipping the leftmost 3 pancakes, getting to ++++-++-, then the rightmost 3, getting to ++++---+, and finally the 3 pancakes that remain blank side up. There are other ways to do it with 3 flips or more, but none with fewer than 3 flips.

In Case #2, all of the pancakes are already happy side up, so there is no need to flip any of them.

In Case #3, there is no way to make the second and third pancakes from the left have the same side up, because any flip flips them both. Therefore, there is no way to make all of the pancakes happy side up.

7. Screenshots of this question will be mailed for you separately.