We-Conect Challenge

PROJECT

Create a simple React Js APP that display a responsive menu. The menu structure will be loaded from a RESTful API by calling GET/api/v1/menu. You can use Mock data.

Example:

```
[
  {
    "slug":"we-conect",
    "title":"We-Conect Home page",
    "url":"https://www.we-conect.com/",

    "children": [
      {
        " slug ":"live-events",
        "title":"Live Event",
        "url":"https://www.we-conect.com/liveevents",
      },
      {
        " slug ":"digital-managed-events ",
        "title":"Digital Managed Events ",
        "url":"https://www.we-conect.com/l digital-managed-events ",
      }
    ]
},
  {
    " slug ":"google",
    "title":"Google",
    "url":"https://www.google.de"
}
]
```

Create a story book for the menu item and use it to create the menu components and make E2E tests using Cypress. When clicking on the menu item, the content of the URL should be displayed inside an internal frame and the path should be the same as the menu item slug.

PROBLEM:

The We-Conect company introduced a new kind of pancake. It has a happy face made of chocolate chips on one side (the "happy side"), and nothing on the other side (the "blank side").

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.

NOTE

Submit the GIT url of your solution to imadedine.jerbi@we-conect.com

Deadline: 3 days.