Data Structure

Assignment 1

Due: March 9, 2022

Submission guideline (Reading carefully):

Your assignment will be graded based on two parts:

(1) Code submission via online judge system

- The system will automatically check whether your code is right or not. If and only if your answer passes all test cases, it will be regarded as the correct answer. Multiple submissions are allowed.
- Submission system: https://vjudge.net/contest/480817#overview (see more details in tutorial#5 slides. Keep remember to set your name as your student ID)

(2) File submission via blackboard

- Submit a pdf file to the blackboard (instead of the scripts).
- The submission should include: (1) code for each question; (2) short paragraph to illustrate your approach for each question.

Question 1 (Perfect Bracket) Brackets are essential in writing, but only the brackets with correct pairs can be compiled in a computer program. Here we use the term "perfect bracket" to denote those correct pairs and define it as follows:

- Empty string is a perfect bracket.
- If s_1 and s_2 is the perfect bracket, then the strings $<s_1>s_2$, $\{s_1\}s_2$, $[s_1]s_2$, $(s_1)s_2$ are also perfect bracket.

Given a string made by opening brackets *<, {, [, (* and closing brackets *>, },],),* we can replace any bracket by another in same type (i.e., opening, closing). For example, *"{"* can be replaced by *"["* or *"("* but not *")" or ">"*. Determine the *smallest number of replacements* to make a string *s* be the perfect bracket.

Examples:

Input with one line of a non-empty string *s,* which is only made up of the brackets. Output the smallest number of replacement to make s be the perfect bracket. If it is impossible to make it a perfect bracket, return "Impossible".

Input:
[<})){}
Ouput:
2
Explanation: Change "[" to "(", and change "<" to "{", respectively.

Input:
((
Output:
Impossible

Question 2 (Ticket Buyer) There are *n* people in a line queuing to buy tickets, where the 1st person is at the front of the line and *n* person is at the back of the line. The ith person would like to buy at least *tickets[i]* number of tickets. A person can only buy **at most *m* ticket at a time** and has to go back to **the end** of the line in order to buy more tickets. To save the waiting time, each person will buy as much as possible tickets at one time. If a person does not have any tickets left to buy, the person will **leave** the line. Return the original position *k* (1-indexed) of the last person in the line.

Examples:

Input with 2 lines, where the first line contains two integers *n, m*, and the second line includes *n* integers *tickets[1], tickets[2], …, tickets[n].*
Output the last person's position *k*.

Input:
6 2
2 3 1 4 2 1
Output:
4
Explanation: In the first pass, the 1st, 3rd,5th, 6th person leave after buying enough tickets and the line becomes [1, 2]. Then the 2nd person buys 1 ticket and leave. The last person is the 4th person.

Question 3 (Inclusive string) We use the term "inclusive string" to indicate a string *s* contains all substrings s_i we need. Given three strings (s_1, s_2, s_3), determine the *minimal length* of an "inclusive string" *s* which contains all three strings.

Examples:

Input with three lines, where each line contains a string s_i *(1<=i<=3).*
Output the minimal length of their inclusive string.

Input:
wx
yz
xy
Output:
4
Explanation: *wxyz* satisfies the requirement of the inclusive string with the length 4.

Chinese University of Hong Kong places very high importance on honesty in academic work submitted by students, and adopts a policy of zero tolerance on academic dishonesty. Any related offence will lead to disciplinary action including termination of studies at the University. Collaboration or discussion of the assignment is allowed, but you need to write down the code and illustration in your own words.