These problems are not directly related to prefix or postfix expression evaluation, but centered around a similar idea of popping items at the top of the stack as we scan through the tokens in the input. Consider this approach when trying to solve the problems.

Problem 1: The Stock Span Problem

The stock span problem is a financial problem where we have a series of n daily price quotes for a stock and we need to calculate the span of stock's price for all n days. The span Si of the stock's price on a given day is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than its price on the given day.

Examples
If an array of 7 days prices is given as `{ 100, 80, 60, 70, 60, 75, 85 }`, then the span values for corresponding 7 days are `{ 1, 1, 1, 2, 1, 4, 6 }`. For example, the stock span on the last day is 6 because the prices on the last 6 days are less than or equal to the price on the last day, i.e. 85.

a. Solve it using 2 loops
Hints
Iterate through the array, and for every item, iterate backwards till you find a stock price value higher than the current value.

b. Solve using 1 loop and a stack
Hints
When iterating through the stock prices, have a stack that maintains indexes of only those items in the stock price array that are larger than the previous day's stock price values.

References

- Theory: https://www.geeksforgeeks.org/the-stock-span-problem/
- Practice: https://practice.geeksforgeeks.org/problems/stock-span-problem-1587115621/1

Problem 2: The Celebrity Problem

In a party of N people, only one person is known to everyone. Such a person may be present in the party, if yes, (s)he doesn't know anyone in the party. We can only ask questions like "does A know B?". Find the stranger (celebrity) in the minimum number of questions.

We can describe the problem input as an array of numbers/characters representing persons in the party. We also have a hypothetical function HaveAcquaintance(A, B) which returns true if A knows B, false otherwise. How can we solve the problem?

Example

```
Input:
MATRIX = {
    {0, 0, 1, 0},
    {0, 0, 1, 0},
    {0, 0, 0, 0},
    {0, 0, 1, 0}
}
Output:id = 2
Explanation: The person with ID 2 does not know anyone but everyone knows him/her
```

Hints

- Choose 2 people (indexes in the array) at random, say a and b. If the value of HaveAcquaintance(A, B) is 0 then B cannot be celebrity. If the value is 1 then A cannot be a celebrity. A stack can initially be formed with all indexes pushed. Then pop two items, eliminate one, and push the remaining back to the stack.

References

- Theory: https://www.geeksforgeeks.org/the-celebrity-problem/
- Practice: https://practice.geeksforgeeks.org/problems/the-celebrity-problem/1

Problem 3: Minimum number of bracket reversals needed to make an expression balanced

Given an expression with only `}` and `{` . The expression may not be balanced. Find the minimum number of bracket reversals to make the expression balanced.

Examples

```
Input:  exp = "}{"
Output: 2
We need to change '}' to '{' and '{' to
'}' so that the expression becomes balanced,
the balanced expression is '{}'


Input:  exp = "{{{"
Output: Can't be made balanced using reversals


Input:  exp = "{{{{"
Output: 2
```

Hints

- First off, only expressions with an even number of brackets have a solution. Then let's say the input expression is `}}{{}}{{}}{{{` . The minimum number of inversions is the same as a simplified expression with the balanced sub-expression removed. That is, the minimum number of inversions of a simplified expression - `}}{{{{` would be the answer (Why this is a so needs some thought centered around how a balanced expression with minimum inversions on the larger expression can be converted step-by-step into a balanced expression for the one with balanced parts intact). We can proceed like we do for brackets matching - i.e. push opening bracket to stack, and pop the last opening bracket when a closing bracket is encountered. Additionally, we push a closing bracket to stack if we do not find a matching opening bracket (i.e. stack is empty).
- If we followed this procedure we would end up with a string of closing brackets followed by a string of opening brackets (like `}}{{{{`). The first bracket must be inverted in any balanced solution. We claim here that one solution with minimum inversions is where we can invert alternate brackets in the string with closing brackets (`}}`) and the string with opening brackets (`{{{{`). This gives a balanced expression, in this example, as `{}{}{}` . With similar though process like before we can prove that any balanced expression with minimum inversions on `}}{{{{` can be translated to the expression `{}{}{}` while maintaining the number of inversions. This process of inverting alternate brackets leads to `Math.ceil(num_opening_brackets / 2) + Math.ceil(num_closing_brackets / 2)`

References

- Theory: https://www.geeksforgeeks.org/minimum-number-of-bracket-reversals-needed-to-make-an-expression-balanced/
- Practice: https://practice.geeksforgeeks.org/problems/count-the-reversals0401/1
- Video: https://www.youtube.com/watch?v=8q1sma-qMsA