

# **Stack**

## **Question - 01**

Implement the push, pop and peek functions using an array.

## **Question - 02**

Implement the push, pop and peek functions using a linked list.

## **Question - 03**

Reverse a string using stack.

Example:

Input: 'CSE220'

Output: '022ESC'

## **Question - 04**

Check whether a string is palindrome or not using stack. If it is palindrome, print True, otherwise, print False.

Example:

Input: "MADAM"

Output: True

Input: "CSE220"

Output: False

### Question - 05

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a queue. Each student either prefers square or circular sandwiches.

The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a **stack**. At each step:

- If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue.
- Otherwise, they will leave it and go to the queue's end.

This continues until none of the queue students want to take the top sandwich and are thus unable to eat.

You are given two integer arrays `students` and `sandwiches` where `sandwiches[i]` is the type of the *i*th sandwich in the stack (*i* = 0 is the top of the stack) and `students[j]` is the preference of the *j*th student in the initial queue (*j* = 0 is the front of the queue). Return the number of students that are unable to eat.

Example 1:

Input: `students = [1,1,0,0]`, `sandwiches = [0,1,0,1]`

Output: 0

Explanation:

- Front student leaves the top sandwich and returns to the end of the line making `students = [1,0,0,1]`.
- Front student leaves the top sandwich and returns to the end of the line making `students = [0,0,1,1]`.
- Front student takes the top sandwich and leaves the line making `students = [0,1,1]` and `sandwiches = [1,0,1]`.
- Front student leaves the top sandwich and returns to the end of the line making `students = [1,1,0]`.
- Front student takes the top sandwich and leaves the line making `students = [1,0]` and `sandwiches = [0,1]`.
- Front student leaves the top sandwich and returns to the end of the line making `students = [0,1]`.
- Front student takes the top sandwich and leaves the line making `students = [1]` and `sandwiches = [1]`.
- Front student takes the top sandwich and leaves the line making `students = []` and `sandwiches = []`.

Hence all students are able to eat.

Example 2:

Input: students = [1,1,1,0,0,1], sandwiches = [1,0,0,0,1,1]

Output: 3

### Question - 06

Convert the following infix notation to postfix using stack following the given precedence of the operators. You must show the workings. You do not need to write code.

Consider the following precedence (decreases down the order)

1.  $*$ ,  $/$ ,  $//$
2.  $\%$
3.  $+$ ,  $-$
4.  $==$ ,  $<=$ ,  $>=$
5.  $\&\&$

Now convert the following infix to its postfix notations:

1.  $A + B * C$
2.  $A * B + C$
3.  $A + B * C - D$
4.  $A + B * (C - D / E)$
5.  $A * (B + C) * D$
6.  $A * B + C * D$
7.  $A - B + C - D * E$
8.  $(A - B + C) * (D + E * F)$
9.  $A * (B + C - (D + E / F))$
10.  $((A + B) - C * (D / E)) + F$

### Question - 07

Convert the following postfix notations to its infix using stack. You must show the workings. You do not need to write code.

1.  $AB -$
2.  $AB + CD + *$
3.  $ABC * + D +$
4.  $AB * CD * +$
5.  $ABC + * D *$
6.  $AB * CD * +$
7.  $AB - C + DE * -$
8.  $AB - C + DEF * + *$
9.  $AB + CDE / * - F +$
10.  $ABCDE / - * +$

### Question - 07

Evaluate the postfix expressions using stack. You must show the workings. You do not need to write code.

1.  $34 * 25 * +$
2.  $23 - 4 + 567 * + *$

Reference:

1. <https://www.geeksforgeeks.org/data-structures/linked-list/doubly-linked-list/>
2. <https://leetcode.com/problems/number-of-students-unable-to-eat-lunch/>
3. <https://web.stonehill.edu/compsci/CS104/Stuff/Infix%20and%20%20postfix%20expressions.pdf>