

# Rajalakshmi Engineering College

Name: Parameswari P  
Email: 240701378@rajalakshmi.edu.in  
Roll no: 240701378  
Phone: 9500133836  
Branch: REC  
Department: I CSE FD  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 3\_CY

Attempt : 1  
Total Mark : 30  
Marks Obtained : 0

### Section 1 : Coding

#### 1. Problem Statement

Rithi is building a simple text editor that allows users to type characters, undo their typing, and view the current text. She has implemented this text editor using an array-based stack data structure.

She has to develop a basic text editor with the following features:

Type a Character (Push): Users can type a character and add it to the text editor. Undo Typing (Pop): Users can undo their typing by removing the last character they entered from the editor. View Current Text (Display): Users can view the current text in the editor, which is the sequence of characters in the buffer. Exit: Users can exit the text editor application.

Write a program that simulates this text editor's undo feature using a character stack and implements the push, pop and display operations accordingly.

### ***Input Format***

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Push the character onto the stack. If the choice is 1, the following input is a space-separated character, representing the character to be pushed onto the stack.

Choice 2: Pop the character from the stack.

Choice 3: Display the characters in the stack.

Choice 4: Exit the program.

### ***Output Format***

The output displays messages according to the choice and the status of the stack:

1. If the choice is 1, print: "Typed character: <character>" where <character> is the character that was pushed to the stack.
2. If the choice is 2, print: "Undo: Removed character <character>" where <character> is the character that was removed from the stack.
3. If the choice is 2, and if the stack is empty without any characters, print "Text editor buffer is empty. Nothing to undo."
4. If the choice is 3, print: "Current text: <character1> <character2> ... <characterN>" where <character1>, <character2>, ... are the characters in the stack, starting from the last pushed character.
5. If the choice is 3, and there are no characters in the stack, print "Text editor buffer is empty."
6. If the choice is 4, exit the program.
7. If any other choice is entered, print "Invalid choice"

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 1 H

1 A

3

4

Output: Typed character: H

Typed character: A

Current text: A H

**Answer**

-

**Status :** -

**Marks :** 0/10

## 2. Problem Statement

You are required to implement a stack data structure using a singly linked list that follows the Last In, First Out (LIFO) principle.

The stack should support the following operations: push, pop, display, and peek.

### **Input Format**

The input consists of four space-separated integers N, representing the elements to be pushed onto the stack.

### **Output Format**

The first line of output displays all four elements in a single line separated by a space.

The second line of output is left blank to indicate the pop operation without displaying anything.

The third line of output displays the space separated stack elements in the same line after the pop operation.

The fourth line of output displays the top element of the stack using the peek operation.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 11 22 33 44

Output: 44 33 22 11

33 22 11

33

**Answer**

-

**Status :** Skipped

**Marks :** 0/10

### 3. Problem Statement

In an educational setting, Professor Smith tasks Computer Science students with designing an algorithm to evaluate postfix expressions efficiently, fostering problem-solving skills and understanding of stack-based computations.

The program prompts users to input a postfix expression, evaluates it, and displays the result, aiding students in honing their coding abilities.

#### **Input Format**

The input consists of the postfix mathematical expression.

The expression will contain real numbers and mathematical operators ( +, -, \*, / ), without any space.

#### **Output Format**

The output prints the result of evaluating the given postfix expression.

Refer to the sample output for formatting specifications.

#### **Sample Test Case**

Input: 82/

Output: 4

**Answer**

-

Status : Skipped

Marks : 0/10