

# Rajalakshmi Engineering College

Name: Nitin Aakash  
Email: 240701370@rajalakshmi.edu.in  
Roll no: 240701370  
Phone: 9498349045  
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\_COD\_Question 1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

In a coding competition, you are assigned a task to create a program that simulates a stack using a linked list.

The program should feature a menu-driven interface for pushing an integer to stack, popping, and displaying stack elements, with robust error handling for stack underflow situations. This challenge tests your data structure skills.

##### ***Input Format***

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

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

the stack.

Choice 2: Pop the integer from the stack.

Choice 3: Display the elements in the stack.

Choice 4: Exit the program.

### ***Output Format***

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

If the choice is 1, push the given integer to the stack and display the following:  
"Pushed element: " followed by the value pushed.

If the choice is 2, pop the integer from the stack and display the following:  
"Popped element: " followed by the value popped.

If the choice is 2, and if the stack is empty without any elements, print "Stack is empty. Cannot pop."

If the choice is 3, print the elements in the stack: "Stack elements (top to bottom): " followed by the space-separated values.

If the choice is 3, and there are no elements in the stack, print "Stack is empty".

If the choice is 4, exit the program and display the following: "Exiting program".

If any other choice is entered, print "Invalid choice".

Refer to the sample input and output for the exact format.

### **Sample Test Case**

Input: 1 3

1 4

3

2

3

4

Output: Pushed element: 3

Pushed element: 4

Stack elements (top to bottom): 4 3

Popped element: 4

Stack elements (top to bottom): 3

Exiting program

### **Answer**

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct Node {  
    int data;  
    struct Node* next;  
};
```

```
struct Node* top = NULL;
```

```
void push(int value) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->data = value;  
    newNode->next = top;  
    top = newNode;  
    printf("Pushed element: %d\n", value);  
}
```

```
void pop() {
```

```

    if (top == NULL) {
        printf("Stack is empty. Cannot pop.\n");
        return;
    }
    struct Node* temp = top;
    int popped = temp->data;
    top = top->next;
    free(temp);
    printf("Popped element: %d\n", popped);
}

```

```

void displayStack() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    printf("Stack elements (top to bottom): ");
    struct Node* curr = top;
    while (curr != NULL) {
        printf("%d ", curr->data);
        curr = curr->next;
    }
    printf("\n");
}

```

```

int main() {
    int choice, value;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d", &value);
                push(value);
                break;
            case 2:
                pop();
                break;
            case 3:
                displayStack();
                break;
            case 4:
                printf("Exiting program\n");
                return 0;
        }
    } while (choice != 4);
}

```

```
        default:
            printf("Invalid choice\n");
        }
    } while (choice != 4);

    return 0;
}
```

**Status :** Correct

**Marks :** 10/10