Lab Experiment: 04

Student Detail:

- **Name:** Prashant Joshi
- **Student ID:** 590010879
- **Branch:** MCA
- **Batch:** B1
- **Instructor:** Dr. Sourbh Kumar

Lab Assignment 1: Stack Implementation Using Arrays

Problem Statement: Implement a stack data structure using an array. Your program should support the following stack operations:

1. Push: Add an element to the top of the stack.
2. Pop: Remove an element from the top of the stack.
3. Peek: Display the top element without removing it.
4. IsEmpty: Check if the stack is empty.
5. IsFull: Check if the stack is full (assume a fixed size).

Assignment Tasks:

- Write a C program that defines a stack using arrays.
- Implement the stack operations mentioned above.
- Demonstrate stack overflow and underflow conditions.
- Write a main program to test all stack operations.

Solution:

```
#include <stdio.h>

#define MAX 5 // Define the maximum size of the stack

int stack[MAX];

int top = -1; // Initial stack is empty

// Function to add an element to the stack

void push(int value) {

   if (top == MAX - 1) {

      printf("Stack Overflow! Cannot push %d\n", value);

   } else {

      stack[++top] = value;

      printf("%d pushed to the stack\n", value);

   }

}

// Function to remove the top element from the stack

void pop() {

   if (top == -1) {

      printf("Stack Underflow! Cannot pop\n");

   } else {
```

```
        printf("%d popped from the stack\n", stack[top--]);
    }
}

// Function to display the top element of the stack
void peek() {
    if (top == -1) {
        printf("Stack is empty\n");
    } else {
        printf("Top element is %d\n", stack[top]);
    }
}

// Function to check if the stack is empty
int isEmpty() {
    return top == -1;
}

// Function to check if the stack is full
int isFull() {
    return top == MAX - 1;
}

// Main function to test the stack operations
int main() {
    int choice, value;

    do {
        printf("\nStack Operations:\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Peek\n");
        printf("4. Check if Empty\n");
```

```
    printf("5. Check if Full\n");
    printf("6. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter value to push: ");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            pop();
            break;
        case 3:
            peek();
            break;
        case 4:
            if (isEmpty()) {
                printf("Stack is empty\n");
            } else {
                printf("Stack is not empty\n");
            }
            break;
        case 5:
            if (isFull()) {
                printf("Stack is full\n");
            } else {
                printf("Stack is not full\n");
            }
            break;
        case 6:
            printf("Exiting program\n");
```

```
            break;
        default:
            printf("Invalid choice! Please try again.\n");
    }
  } while (choice != 6);

  return 0;
}
```

Output:

```
Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Check if Full
6. Exit
Enter your choice: 1
Enter value to push: 12
12 pushed to the stack

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Check if Full
6. Exit
Enter your choice: 2
12 popped from the stack
```

Lab Assignment 2:

Problem Statement: Implement a stack data structure using a linked list. The program should support the following operations:

1. Push: Add an element to the top of the stack.
2. Pop: Remove an element from the top of the stack.
3. Peek: Display the top element without removing it.
4. IsEmpty: Check if the stack is empty.

Assignment Tasks:

- Write a C program that defines a stack using a singly linked list.
- Implement the stack operations mentioned above.
- Demonstrate stack operations using linked lists.
- Write a main program to test all stack operations.

Solution:

```
#include <stdio.h>
#include <stdlib.h>
// Define a node structure for the stack
struct Node {
    int data;
    struct Node* next;
};
// Top of the stack
struct Node* top = NULL;
// Function to check if the stack is empty
int isEmpty() {
    return top == NULL;
}
// Function to add an element to the top of the stack
void push(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Heap Overflow! Cannot push %d\n", value);
        return;
    }
```

```
    newNode->data = value;
    newNode->next = top;
    top = newNode;
    printf("%d pushed to the stack\n", value);
}

// Function to remove the top element from the stack
void pop() {
    if (isEmpty()) {
        printf("Stack Underflow! Cannot pop\n");
        return;
    }
    struct Node* temp = top;
    printf("%d popped from the stack\n", top->data);
    top = top->next;
    free(temp);
}

// Function to display the top element of the stack
void peek() {
    if (isEmpty()) {
        printf("Stack is empty\n");
    } else {
        printf("Top element is %d\n", top->data);
    }
}

// Main function to test the stack operations
int main() {
    int choice, value;

    do {
        printf("\nStack Operations:\n");
```

```
    printf("1. Push\n");
    printf("2. Pop\n");
    printf("3. Peek\n");
    printf("4. Check if Empty\n");
    printf("5. Exit\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            printf("Enter value to push: ");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            pop();
            break;
        case 3:
            peek();
            break;
        case 4:
            if (isEmpty()) {
                printf("Stack is empty\n");
            } else {
                printf("Stack is not empty\n");
            }
            break;
        case 5:
            printf("Exiting program\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
    }
```

```
    } while (choice != 5);

    return 0;
}
```

Output:

```
Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice: 1
Enter value to push: 223
223 pushed to the stack

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice: 3
Top element is 223

Stack Operations:
1. Push
2. Pop
3. Peek
4. Check if Empty
5. Exit
Enter your choice: 5
Exiting program
```