Lab Experiment: 05

Student Detail:

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

Lab Assignment 1: Queue Implementation Using Arrays

Problem Statement:

Implement a queue data structure using arrays. Your program should support the following queue operations:

1. Enqueue: Add an element to the rear of the queue.
2. Dequeue: Remove an element from the front of the queue.
3. Peek: Display the front element without removing it.
4. IsEmpty: Check if the queue is empty.
5. IsFull: Check if the queue is full (assume a fixed size).

Assignment Tasks:

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

Solution:

```
#include <stdio.h>

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


int queue[MAX];

int front = -1, rear = -1;


// Function to check if the queue is empty

int isEmpty() {

   return front == -1;

}


// Function to check if the queue is full

int isFull() {

   return rear == MAX - 1;

}


// Function to add an element to the rear of the queue

void enqueue(int value) {
```

```
    if (isFull()) {
        printf("Queue Overflow! Cannot enqueue %d\n", value);
    } else {
        if (front == -1) front = 0;
        queue[++rear] = value;
        printf("%d enqueued to the queue\n", value);
    }
}

// Function to remove an element from the front of the queue
void dequeue() {
    if (isEmpty()) {
        printf("Queue Underflow! Cannot dequeue\n");
    } else {
        printf("%d dequeued from the queue\n", queue[front]);
        if (front == rear) {
            front = rear = -1; // Reset the queue if it's empty
        } else {
            front++;
        }
    }
}

// Function to display the front element of the queue
void peek() {
    if (isEmpty()) {
        printf("Queue is empty\n");
    } else {
        printf("Front element is %d\n", queue[front]);
    }
}

// Main function to test the queue operations
```

```
int main() {
    int choice, value;

    do {
        printf("\nQueue Operations:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\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 enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                peek();
                break;
            case 4:
                if (isEmpty()) {
                    printf("Queue is empty\n");
                } else {
                    printf("Queue is not empty\n");
                }
                break;
```

```
            case 5:
                if (isFull()) {
                    printf("Queue is full\n");
                } else {
                    printf("Queue 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:

```
Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Empty
5. Check if Full
6. Exit
Enter your choice: 1
Enter value to enqueue: 123
123 enqueued to the queue

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Empty
5. Check if Full
6. Exit
Enter your choice: 3
Front element is 123
```

Lab Assignment 2: Queue Implementation Using Linked Lists

Problem Statement:

Implement a queue data structure using a linked list. Your program should support the following operations:

1. Enqueue: Add an element to the rear of the queue.
2. Dequeue: Remove an element from the front of the queue.
3. Peek: Display the front element without removing it.
4. IsEmpty: Check if the queue is empty.

Assignment Tasks:

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

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

// Define a node structure for the queue
struct Node {
    int data;
    struct Node* next;
};

// Front and rear of the queue
struct Node* front = NULL;
struct Node* rear = NULL;

// Function to check if the queue is empty
int isEmpty() {
    return front == NULL;
}

// Function to add an element to the rear of the queue
void enqueue(int value) {
```

```
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Heap Overflow! Cannot enqueue %d\n", value);
        return;
    }
    newNode->data = value;
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
    printf("%d enqueued to the queue\n", value);
}

// Function to remove an element from the front of the queue
void dequeue() {
    if (isEmpty()) {
        printf("Queue Underflow! Cannot dequeue\n");
        return;
    }
    struct Node* temp = front;
    printf("%d dequeued from the queue\n", front->data);
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
}

// Function to display the front element of the queue
void peek() {
```

```
    if (isEmpty()) {
        printf("Queue is empty\n");
    } else {
        printf("Front element is %d\n", front->data);
    }
}

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

    do {
        printf("\nQueue Operations:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\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 enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                peek();
                break;
```

```
            case 4:
                if (isEmpty()) {
                    printf("Queue is empty\n");
                } else {
                    printf("Queue 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:

```
Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Empty
5. Exit
Enter your choice: 1
Enter value to enqueue: 342
342 enqueued to the queue

Queue Operations:
1. Enqueue
2. Dequeue
3. Peek
4. Check if Empty
5. Exit
Enter your choice: 1
Enter value to enqueue: 56
56 enqueued to the queue
```