1. **Implementation of Queue Operations by circular array**
    a. Insert elements in the queue.
    b. Delete elements from the queue.
    c. Print the queue front element.

Program –

```
#include <stdio.h>
#define MAX 5
int cqueue_arr[MAX];
int front = -1;
int rear = -1;
void insert(int item)
{
    if ((front == 0 && rear == MAX - 1) || (front == rear
+ 1))
    {
        printf("Queue Overflow \n");
        return;
    }
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }
    else
    {
        if (rear == MAX - 1)
            rear = 0;
        else
            rear = rear + 1;
    }
    cqueue_arr[rear] = item;
}
void deletion()
{
    if (front == -1)
    {
        printf("Queue Underflown\n");
        return;
    }
    printf("Element deleted from queue is : %d\n",
cqueue_arr[front]);
```

```
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else
    {
        if (front == MAX - 1)
            front = 0;
        else
            front = front + 1;
    }
}
void display()
{
    int front_pos = front, rear_pos = rear;
    if (front == -1)
    {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements:");
    if (front_pos <= rear_pos)
        while (front_pos <= rear_pos)
        {
            printf("%d ", cqueue_arr[front_pos]);
            front_pos++;
        }
    else
    {
        while (front_pos <= MAX - 1)
        {
            printf("%d ", cqueue_arr[front_pos]);
            front_pos++;
        }
        front_pos = 0;
        while (front_pos <= rear_pos)
        {
            printf("%d ", cqueue_arr[front_pos]);
            front_pos++;
```

```
        }
    }
    printf("\n");
}
int main()
{
    int choice, item;
    do
    {
        printf("1.Insert\n");
        printf("2.Delete\n");
        printf("3.Display\n");
        printf("4.Quit\n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Input the element for insertion in
queue : ");
            scanf("%d", &item);
            insert(item);
            break;
        case 2:
            deletion();
            break;
        case 3:
            display();
            break;
        case 4:
            break;
        default:
            printf("Wrong choicen");
        }
    } while (choice != 4);
    return 0;
}
```

2. Implementation of Queue Operations by Linked List

a. Insert elements in the queue.
b. Delete elements from the queue.

Program –

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
    int choice;
    while(choice != 4)
    {
        printf("\n1. Insert an element\n2. Delete an
element\n3. Display the queue\n4. Exit\n");
        printf("\nEnter your choice? ");
        scanf("%d",& choice);
        switch(choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(0);
            break;
            default:
```

```
            printf("Enter valid choice??\n");
        }
    }
}
void insert()
{
    struct node *ptr;
    int item;

    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("OVERFLOW\n");
        return;
    }
    else
    {
        printf("Enter value: ");
        scanf("%d",&item);
        ptr -> data = item;
        if(front == NULL)
        {
            front = ptr;
            rear = ptr;
            front -> next = NULL;
            rear -> next = NULL;
        }
        else
        {
            rear -> next = ptr;
            rear = ptr;
            rear->next = NULL;
        }
    }
}
void delete ()
{
    struct node *ptr;
    if(front == NULL)
    {
```

```
        printf("UNDERFLOW\n");
        return;
    }
    else
    {
        ptr = front;
        front = front -> next;
        free(ptr);
    }
}
void display()
{
    struct node *ptr;
    ptr = front;
    if(front == NULL)
    {
        printf("Empty queue\n");
    }
    else
    {   printf("Printing values: ");
        while(ptr != NULL)
        {
            printf("%d ",ptr -> data);
            ptr = ptr -> next;
        }
    }
    printf("\n");
}
```