Implementation of Queue Using Linked

Aim

To write and execute a C program to **implement a queue using a linked list** .

THEORY

A **Queue** is a linear data structure that follows the **FIFO (First In First Out)** principle.Queue can be implemented using arrays or linked lists. Queue implementation using arrays is efficient when only **one queue** is required.However, when **multiple queues** need to coexist, there is no efficient way to implement them sequentially. A good solution to this problem is to use Linked List to implement Queue. Such a Queue is called Linked Queue. In this algorithm we can easily add a node at the rear end and delete from the front end of the linked queue.

- **Insertion (ENQUEUE)** is performed at the **rear end** of the queue.
- **Deletion (DEQUEUE)** is performed at the **front end** of the queue.

Linked queue allows **dynamic memory allocation**, so the queue can grow or shrink as needed.Queue overflow occurs only when system memory is full.

Program

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

/* Structure of a node */
struct node {
   int data;
   struct node *next;
};

struct node *front = NULL;
struct node *rear = NULL;

/* Function declarations */
void enqueue();
void dequeue();
void display();

int main() {
   int choice;

   do {
      printf("\n\n--- QUEUE MENU ---");
      printf("\n1. Enqueue");
      printf("\n2. Dequeue");
      printf("\n3. Display");
```

```
        printf("\n4. Exit");

        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: enqueue(); break;
            case 2: dequeue(); break;
            case 3: display(); break;
            case 4: printf("Exiting program"); break;
            default: printf("Invalid choice");
        }
    } while (choice != 4);

    return 0;
}

/* Enqueue operation */
void enqueue() {
    struct node *newnode;
    int value;

    newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Queue Overflow (Memory not allocated)");
        return;
    }

    printf("Enter value to insert: ");
    scanf("%d", &value);

    newnode->data = value;
    newnode->next = NULL;

    if (rear == NULL) {
        front = rear = newnode;
    } else {
        rear->next = newnode;
        rear = newnode;
    }

    printf("Element inserted into queue");
}

/* Dequeue operation */
void dequeue() {
    struct node *temp;

    if (front == NULL) {
        printf("Queue Underflow");
```

```
        return;
    }

    temp = front;
    printf("Deleted element: %d", temp->data);
    front = front->next;

    if (front == NULL)
        rear = NULL;

    free(temp);
}

/* Display operation */
void display() {
    struct node *temp;

    if (front == NULL) {
        printf("Queue is empty");
        return;
    }

    temp = front;
    printf("Queue elements:\n");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}
```