

**Data Structure and Algorithm (MCA 271)**

**Lab Practical –**

***BY***

**Himanshu Heda (24225013)**

**SUBMITTED TO**

**Prof. Vandna Kansal**

**SCHOOL OF SCIENCES**

**2024-2025**

**Program Description:**

**Code of the program**

**Output**: - Paste the o/p of the program.

#include <stdio.h>

#include <stdlib.h>

// Node structure for the linked list

struct Node {

    int data;

    struct Node\* next;

};

// Queue structure

struct Queue {

    struct Node\* front;

    struct Node\* rear;

};

// Function to create a new node

struct Node\* createNode(int data) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->next = NULL;

    return newNode;

}

// Function to initialize the queue

struct Queue\* createQueue() {

    struct Queue\* queue = (struct Queue\*)malloc(sizeof(struct Queue));

    queue->front = queue->rear = NULL;

    return queue;

}

// Function to check if the queue is empty

int isEmpty(struct Queue\* queue) {

    return queue->front == NULL;

}

// Function to add an item to the queue

void enqueue(struct Queue\* queue, int data) {

    struct Node\* newNode = createNode(data);

    if (queue->rear == NULL) {

        queue->front = queue->rear = newNode;

        return;

    }

    queue->rear->next = newNode;

    queue->rear = newNode;

}

// Function to remove an item from the queue

int dequeue(struct Queue\* queue) {

    if (isEmpty(queue)) {

        printf("Queue is empty!\n");

        return -1; // Indicate that the queue is empty

    }

    struct Node\* temp = queue->front;

    int data = temp->data;

    queue->front = queue->front->next;

    if (queue->front == NULL) {

        queue->rear = NULL;

    }

    free(temp);

    return data;

}

// Function to display the queue

void display(struct Queue\* queue) {

    if (isEmpty(queue)) {

        printf("Queue is empty!\n");

        return;

    }

    struct Node\* temp = queue->front;

    printf("Queue elements: ");

    while (temp != NULL) {

        printf("%d ", temp->data);

        temp = temp->next;

    }

    printf("\n");

}

// Main function to demonstrate queue operations

int main() {

    struct Queue\* queue = createQueue();

    int choice, value;

    do {

        printf("\nQueue Operations using Linked List\n");

        printf("1. Enqueue\n");

        printf("2. Dequeue\n");

        printf("3. Display\n");

        printf("4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter value to enqueue: ");

                scanf("%d", &value);

                enqueue(queue, value);

                break;

            case 2:

                value = dequeue(queue);

                if (value != -1) {

                    printf("Dequeued: %d\n", value);

                }

                break;

            case 3:

                display(queue);

                break;

            case 4:

                printf("Exiting...\n");

                break;

            default:

                printf("Invalid choice! Please try again.\n");

        }

    } while (choice != 4);

    // Free the allocated memory

    while (!isEmpty(queue)) {

        dequeue(queue);

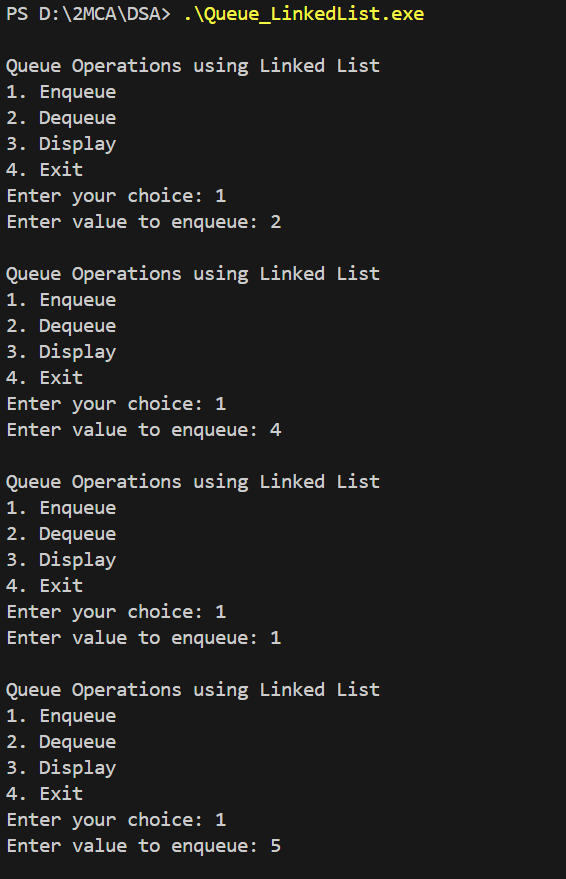
    }

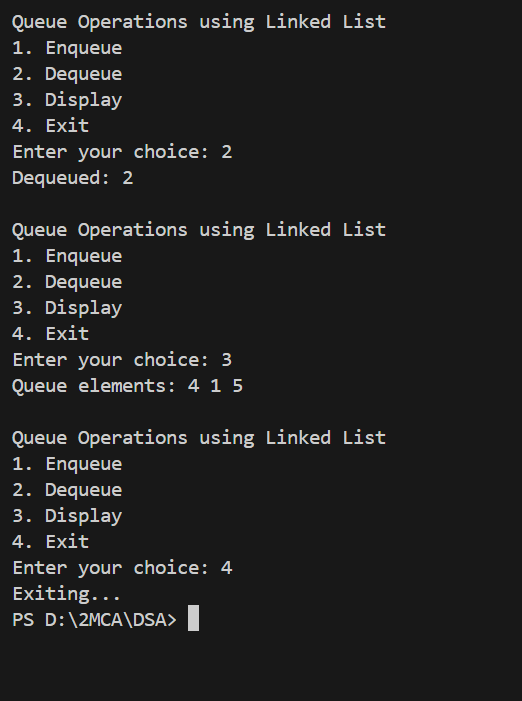
    free(queue);

    return 0;

}

**OUTPUT : --**

****

****