**Aim:- Write a program to implement queue using linked list.**

**// A C program to demonstrate linked list based implementation of queue**

**#include <stdio.h>**

**#include <stdlib.h>**

**// A linked list (LL) node to store a queue entry**

**struct QNode {**

**int key;**

**struct QNode\* next;**

**};**

**// The queue, front stores the front node of LL and rear stores the**

**// last node of LL**

**struct Queue {**

**struct QNode \*front, \*rear;**

**};**

**// A utility function to create a new linked list node.**

**struct QNode\* newNode(int k)**

**{**

**struct QNode\* temp = (struct QNode\*)malloc(sizeof(struct QNode));**

**temp->key = k;**

**temp->next = NULL;**

**return temp;**

**}**

**// A utility function to create an empty queue**

**struct Queue\* createQueue()**

**{**

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

**q->front = q->rear = NULL;**

**return q;**

**}**

**// The function to add a key k to q**

**void enQueue(struct Queue\* q, int k)**

**{**

**// Create a new LL node**

**struct QNode\* temp = newNode(k);**

**// If queue is empty, then new node is front and rear both**

**if (q->rear == NULL) {**

**q->front = q->rear = temp;**

**return;**

**}**

**// Add the new node at the end of queue and change rear**

**q->rear->next = temp;**

**q->rear = temp;**

**}**

**// Function to remove a key from given queue q**

**void deQueue(struct Queue\* q)**

**{**

**// If queue is empty, return NULL.**

**if (q->front == NULL)**

**return;**

**// Store previous front and move front one node ahead**

**struct QNode\* temp = q->front;**

**q->front = q->front->next;**

**// If front becomes NULL, then change rear also as NULL**

**if (q->front == NULL)**

**q->rear = NULL;**

**free(temp);**

**}**

**// Driver Program to test anove functions**

**int main()**

**{**

**struct Queue\* q = createQueue();**

**enQueue(q, 10);**

**enQueue(q, 20);**

**deQueue(q);**

**deQueue(q);**

**enQueue(q, 30);**

**enQueue(q, 40);**

**enQueue(q, 50);**

**deQueue(q);**

**printf("Queue Front : %d \n", q->front->key);**

**printf("Queue Rear : %d", q->rear->key);**

**return 0;**

**}**

**Output**

Queue Front : 40

Queue Rear : 50