**Aim:** Implement Linear Queue ADT using Linked List.

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

typedef struct Node { int data;

struct Node\* next;

} Node;

Node\* createNode(int new\_data)

{

Node\* new\_node = (Node\*)malloc(sizeof(Node)); new\_node->data = new\_data;

new\_node->next = NULL; return new\_node;

}

typedef struct Queue {

Node \*front, \*rear;

} Queue;

Queue\* createQueue()

{

Queue\* q = (Queue\*)malloc(sizeof(Queue)); q->front = q->rear = NULL;

return q;

}

int isEmpty(Queue\* q)

{

if (q->front == NULL && q->rear == NULL) { return 1;

}

return 0;

}

void enqueue(Queue\* q, int new\_data)

{

Node\* new\_node = createNode(new\_data);

if (q->rear == NULL) {

q->front = q->rear = new\_node; return;

}

q->rear->next = new\_node; q->rear = new\_node;

}

void dequeue(Queue\* q)

{

if (isEmpty(q)) {

printf("Queue Underflow\n"); return;

}

Node\* temp = q->front;

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

if (q->front == NULL) q->rear = NULL;

free(temp);

}

int getFront(Queue\* q)

{

if (isEmpty(q)) { printf("Queue is empty\n"); return INT\_MIN;

}

return q->front->data;

}

int getRear(Queue\* q)

{

if (isEmpty(q)) {

printf("Queue is empty\n"); return INT\_MIN;

}

return q->rear->data;

}

int main()

{

Queue\* q = createQueue();

enqueue(q, 10);

enqueue(q, 20);

printf("Queue Front: %d\n", getFront(q)); printf("Queue Rear: %d\n", getRear(q));

dequeue(q);

dequeue(q);

enqueue(q, 30);

enqueue(q, 40);

enqueue(q, 50);

dequeue(q);

printf("Queue Front: %d\n", getFront(q)); printf("Queue Rear: %d\n", getRear(q)); return 0;

}