**Date: 12/01/2023**

**USN: 1NT21IS092**

**Design and implement a given type of (ordinary queue, circular queue) queue in C (array implementation/ Linked list implementation). And demonstrate its working with suitable inputs. Display appropriate messages in case of exceptions**

AIM:- Implementation of queue using linked list.

**ALGORITHIM:-**

**For Insert operation:**

**Step 1**: Allocate the space for the new node PTR

**Step 2**: SET PTR -> DATA = VAL

**Step 3**: IF FRONT = NULL

SET FRONT = REAR = PTR

SET FRONT -> NEXT = REAR -> NEXT = NULL

ELSE

SET REAR -> NEXT = PTR

SET REAR = PTR

SET REAR -> NEXT = NULL

[END OF IF]

**Step 4**: END

**Deletion**

**Step 1**: IF FRONT = NULL

Write " Underflow "

Go to Step 5

[END OF IF]

**Step 2**: SET PTR = FRONT

**Step 3**: SET FRONT = FRONT -> NEXT

**Step 4**: FREE PTR

**Step 5**: END

**PROGRAM-**

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

#include <stdio.h>

#include <stdlib.h>

struct Node { // A linked list (LL) node to store a queue entry

int data;

struct Node\* next;

};

struct Queue { // The queue, front stores the front node of LL and rear stores the last node of LL

struct Node \*front, \*rear;

};

struct Node\* newNode(int k) // A utility function to create a new linked list node.

{

struct Node\* temp

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

temp->data = k;

temp->next = NULL;

return temp;

}

struct Queue\* createQueue() // A utility function to create an empty queue

{

struct Queue\* q

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

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

return q;

}

void enQueue(struct Queue\* q, int k) // The function to add a key k to q

{

struct Node\* temp = newNode(k); // Create a new LL node

if (q->rear == NULL) { // If queue is empty, then new node is front and rear both

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

return;

}

q->rear->next = temp; // Add the new node at the end of queue and change rear

q->rear = temp;

}

void deQueue(struct Queue\* q) // Function to remove a key from given queue q

{

if (q->front == NULL) // If queue is empty, return NULL.

return;

struct Node\* temp = q->front; // Store previous front and move front one node ahead

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

if (q->front == NULL) // If front becomes NULL, then change rear also as NULL

q->rear = NULL;

free(temp);

}

// Main code

int main()

{

struct Queue\* q = createQueue();

enQueue(q, 10);

enQueue(q, 20);

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

printf("Queue Rear : %d\n", q->rear->data);

deQueue(q);

deQueue(q);

enQueue(q, 30);

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

printf("Queue Rear : %d\n", q->rear->data);

enQueue(q, 40);

enQueue(q, 50);

deQueue(q);

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

printf("Queue Rear : %d\n", q->rear->data);

return 0;

}

**OUTPUT:**

