**C program to implement circular queue using SLL.**

//C++ program for insertion and

// deletion in Circular Queue

#include <bits/stdc++.h>

using namespace std;

// Structure of a Node

struct Node

{

int data;

struct Node\* link;

};

struct Queue

{

struct Node \*front, \*rear;

};

// Function to create Circular queue

void enQueue(Queue \*q, int value)

{

struct Node \*temp = new Node;

temp->data = value;

if (q->front == NULL)

q->front = temp;

else

q->rear->link = temp;

q->rear = temp;

q->rear->link = q->front;

}

// Function to delete element from Circular Queue

int deQueue(Queue \*q)

{

if (q->front == NULL)

{

printf ("Queue is empty");

return INT\_MIN;

}

// If this is the last node to be deleted

int value; // Value to be dequeued

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

{

value = q->front->data;

free(q->front);

q->front = NULL;

q->rear = NULL;

}

else // There are more than one nodes

{

struct Node \*temp = q->front;

value = temp->data;

q->front = q->front->link;

q->rear->link= q->front;

free(temp);

}

return value ;

}

// Function displaying the elements of Circular Queue

void displayQueue(struct Queue \*q)

{

struct Node \*temp = q->front;

printf("\nElements in Circular Queue are: ");

while (temp->link != q->front)

{

printf("%d ", temp->data);

temp = temp->link;

}

printf("%d", temp->data);

}

/\* Driver of the program \*/

int main()

{

// Create a queue and initialize front and rear

Queue \*q = new Queue;

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

// Inserting elements in Circular Queue

enQueue(q, 14);

enQueue(q, 22);

enQueue(q, 6);

// Display elements present in Circular Queue

displayQueue(q);

// Deleting elements from Circular Queue

printf("\nDeleted value = %d", deQueue(q));

printf("\nDeleted value = %d", deQueue(q));

// Remaining elements in Circular Queue

displayQueue(q);

enQueue(q, 9);

enQueue(q, 20);

displayQueue(q);

return 0;

}

