JASKIRAT SINGH (2020CSC1008)

Code

```
//C++ program to implement the circular queue using double linked list
#include <iostream>
using namespace std;

//Node class of doubly circular linked list
template <typename T>
class Node
{
public:
   T data;
   Node *next;
   Node *prev;
};

//circular queue class
template <typename T>
class CircularQueue
{
private:
   Node<T> *front;
   Node<T> *rear;

public:
   CircularQueue();
   Node<T> *createNode(T data);
   void enqueue(T data);
   void dequeue();
   Node<T> *peek();
   void display();
};

template <typename T>
CircularQueue<T>::CircularQueue()
{
   front = NULL;
```

```
    rear = NULL;
}

//method to create node
template <typename T>
Node<T> *CircularQueue<T>::createNode(T data)
{
    Node<T> *temp = new Node<T>;
    temp->data = data;
    temp->next = NULL;
    temp->prev = NULL;
    return temp;
}

//method to enquque element in the circular queue
template <typename T>
void CircularQueue<T>::enqueue(T data)
{
    Node<T> *newnode = createNode(data);

    //if queue is empty
    if (rear == NULL)
    {
        cout << "\nEnqueued Element: " << newnode->data << endl;
        front = newnode;
        rear = newnode;
        front->next = front;
        front->prev = front;
    }

    //else- queue already contains some node
    else
    {
        cout << "\nEnqueued Element: " << newnode->data << endl;
        rear->next = newnode;
        newnode->prev = rear;
        newnode->next = front;
        front->prev = newnode;
```

```
        rear = newnode;
    }
}

//method to enquque element in the circular dequeue
template <typename T>
void CircularQueue<T>::dequeue()
{
    Node<T> *temp = front;

    //if queue is empty
    if (front == NULL && rear == NULL)
    {
        cout << "\nQueue Underflow" << endl;
    }

    //if queue has only one node
    else if (front == rear)
    {
        cout << "\nDequeued Element: " << front->data << endl;
        front = NULL;
        rear = NULL;
        delete temp;
    }

    //else- queue has multiple nodes
    else
    {
        cout << "\nDequeued Element: " << front->data << endl;
        front = front->next;
        front->prev = rear;
        rear->next = front;
        delete temp;
    }
}

//method ot return the peek element
template <typename T>
```

```
Node<T> *CircularQueue<T>::peek()
{
   //if queue is empty
   if (front == NULL && rear == NULL)
   {
      cout << "\nQueue Underflow" << endl;
      return NULL;
   }

   //else- queue has multiple elements-return the front
   else
   {
      cout << "\nElement at Peek: " << front->data << endl;
   }
   return front;
}

//method to display all the elements of circular queue
template <typename T>
void CircularQueue<T>::display()
{
   Node<T> *temp = front;

   //queue is empty
   if (front == NULL && rear == NULL)
   {
      cout << "\nQueue is empty" << endl;
   }
   else
   {
      cout << "\nElements in teh circular queue: ";
      do
      {
         cout << " " << temp->data;
         temp = temp->next;
      } while (temp != front);
      cout << endl;
   }
```

```
}

//driver code
int main()
{
    cout << "\n|***|Program Started|***|" << endl;

    CircularQueue<int> q;
    //enqueuing elements in circular queue
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);

    //displaying circular queue
    q.display();

    Node<int> *peek = q.peek();
    //dequeuing elements from circular queue
    q.dequeue();
    //displaying circular queue
    q.display();

    peek = q.peek();
    //dequeuing elements from circular queue
    q.dequeue();
    //displaying circular queue
    q.display();

    peek = q.peek();
    //dequeuing elements from circular queue
    q.dequeue();
    //displaying circular queue
    q.display();
    peek = q.peek();

    cout << "\n|***|Program Ended|***|" << endl;

    return 0;
```

}

Output

```
|***|Program Started|***|

Enqueued Element: 1

Enqueued Element: 2

Enqueued Element: 3

Elements in teh circular queue:  1 2 3

Element at Peek: 1

Dequeued Element: 1

Elements in teh circular queue:  2 3

Element at Peek: 2

Dequeued Element: 2

Elements in teh circular queue:  3

Element at Peek: 3

Dequeued Element: 3

Queue is empty

Queue Underflow

|***|Program Ended|***|
```