JASKIRAT SINGH (2020CSC1008)

Code

```
// C++ program to implement Double ended queue using array
#include <iostream>
using namespace std;

// Maximum size of array or Deque
#define MAX 100

// A structure to represent a Deque
class Deque
{
    int arr[MAX];
    int front;
    int rear;
    int size;

public:
    Deque(int size)
    {
        front = -1;
        rear = 0;
        this->size = size;
    }

    // Operations on Deque:
    void insertfront(int key);
    void insertrear(int key);
    void deletefront();
    void deleterear();
    bool isFull();
    bool isEmpty();
    int getFront();
    int getRear();
    void display();
```

```
};

// Checks whether Deque is full or not.
bool Deque::isFull()
{
    return ((front == 0 && rear == size - 1) ||
            front == rear + 1);
}

// Checks whether Deque is empty or not.
bool Deque::isEmpty()
{
    return (front == -1);
}

// Inserts an element at front
void Deque::insertfront(int key)
{
    // check whether Deque if full or not
    if (isFull())
    {
        cout << "Overflow\n"
             << endl;
        return;
    }

    // If queue is initially empty
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }

    // front is at first position of queue
    else if (front == 0)
        front = size - 1;

    else // decrement front end by '1'
```

```
        front = front - 1;

    // insert current element into Deque
    arr[front] = key;
}

// function to inset element at rear end of Deque.
void Deque ::insertrear(int key)
{
    if (isFull())
    {
        cout << " Overflow\n " << endl;
        return;
    }

    // If queue is initially empty
    if (front == -1)
    {
        front = 0;
        rear = 0;
    }

    // rear is at last position of queue
    else if (rear == size - 1)
        rear = 0;

    // increment rear end by '1'
    else
        rear = rear + 1;

    // insert current element into Deque
    arr[rear] = key;
}

// Deletes element at front end of Deque
void Deque ::deletefront()
{
    // check whether Deque is empty or not
```

```
    if (isEmpty())
    {
        cout << "Queue Underflow\n"
             << endl;
        return;
    }

    // Deque has only one element
    if (front == rear)
    {
        front = -1;
        rear = -1;
    }
    else
        // back to initial position
        if (front == size - 1)
        front = 0;

    else // increment front by '1' to remove current
        // front value from Deque
        front = front + 1;
}

// Delete element at rear end of Deque
void Deque::deleterear()
{
    if (isEmpty())
    {
        cout << " Underflow\n"
             << endl;
        return;
    }

    // Deque has only one element
    if (front == rear)
    {
        front = -1;
        rear = -1;
```

```
    }
    else if (rear == 0)
        rear = size - 1;
    else
        rear = rear - 1;
}

// Returns front element of Deque
int Deque::getFront()
{
    // check whether Deque is empty or not
    if (isEmpty())
    {
        cout << " Underflow\n"
             << endl;
        return -1;
    }
    return arr[front];
}

// function return rear element of Deque
int Deque::getRear()
{
    // check whether Deque is empty or not
    if (isEmpty() || rear < 0)
    {
        cout << " Underflow\n"
             << endl;
        return -1;
    }
    return arr[rear];
}

//functions to display all the element of Deque
void Deque::display()
{
    cout << "Elements in the deque : ";
    if (rear >= front)
```

```
    {
        for (int i = front; i <= rear; i++)
            cout << arr[i] << " ";
    }
    else
    {
        for (int i = front; i < size; i++)
            cout << arr[i] << " ";
    }
    cout << endl;
}

// Driver program to test above function
int main()
{

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

    Deque dq(5);
    cout << "Insert element at rear end : 5 \n";
    dq.insertrear(5);
    cout << endl;
    dq.display();
    cout << endl;

    cout << "insert element at rear end : 10 \n";
    dq.insertrear(10);
    cout << endl;
    dq.display();
    cout << endl;

    cout << "get rear element "
        << " "
        << dq.getRear() << endl;

    dq.deleterear();
    cout << "After delete rear element new rear"
        << " become " << dq.getRear() << endl;
```

```
    cout << endl;
    dq.display();
    cout << endl;

    cout << "inserting element at front end \n";
    dq.insertfront(15);
    cout << endl;
    dq.display();
    cout << endl;

    cout << "get front element "
        << " "
        << dq.getFront() << endl;

    dq.deletefront();

    cout << "After delete front element new "
        << "front become " << dq.getFront() << endl;
    cout << endl;
    dq.display();
    cout << endl;

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

    return 0;
}
```

Output

```
|***|Program Started|***|
Insert element at rear end : 5

Elements in the deque : 5

insert element at rear end : 10

Elements in the deque : 5 10

get rear element  10
After delete rear element new rear become 5

Elements in the deque : 5

inserting element at front end

Elements in the deque : 15

get front element  15
After delete front element new front become 5

Elements in the deque : 5
```