Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering

Data Structures & Algorithms Lab (DJ19ECSBL1)

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Experiment no: 3 Date: 6-10-2022

Aim: Write a program to implement linear and circular queue.

Programming Language: C/C++/Java

Theory: A linear queue is a linear data structure that serves the request first, which has been arrived first. It consists of data elements which are connected in a linear fashion. It has two pointers, i.e., front and rear, where the insertion takes place from the front end, and deletion occurs from the front end.

A Circular Queue is a special version of linear queue where the last element of the queue is connected to the first element of the queue forming a circle. The operations are performed based on FIFO (First In First Out) principle. It is also called **'Ring Buffer'**.

Linear Queue:

Algorithm for Enqueue(Insertion) operation:

Step 1 - Check whether queue is FULL. i.e. (rear == MAX-1)

If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate the function.

Step 2:If it is first element of queue, then set front=rear=0
Step 3 - If it is NOT first element, then increment rear value by one (rear++)
Step 4 -set queue[rear] = value
Step 5: Exit

Algorithm for Dequeue (Deletion) operation:

Step 1 - Check whether queue is EMPTY. (front==-1)

If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminate the function.

Step 2 - If it is NOT EMPTY, then print the front value i.e. queue[front] to be deleted.
Step 3-Check the value of front pointer. If it is equal to rear it means it is the last element of the queue. The queue will become empty after deleting this element. In this case set front and rear both pointers to -1.
Step 4: Else increment the front value by one (front ++).
Step 5: Exit

Algorithm for Display operation:

Step 1 - Check whether queue is EMPTY. (front == -1)

If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.

Step 2 - If it is NOT EMPTY, then traverse the queue from front position to rear position. Define an integer variable 'i' and set 'i = front'.Display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i' value reaches to rear (i <= rear)

Step 3: Exit

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Circular Queue:

Algorithm for Enqueue(Insertion) operation:

Step 1 - Check whether queue is FULL. ((rear == SIZE-1 && front == 0) || (front == rear+1))

- If it is FULL, then display "Queue is FULL!!! Insertion is not possible!!!" and terminate the function.

Step 2: Else if inserted element is first element of the queue then set front=rear=0

Step 3 - If space is there in the array towards LHS (rear == SIZE - 1 && front != 0), then implement queue in circular fashion set rear = 0.

Step 4 - else, Increment rear value by one (rear++)

Step 5 - set queue[rear] = value

Step 6: Exit

Algorithm for Dequeue (Deletion) operation:

Step 1 - Check whether queue is EMPTY. (front == -1 && rear == -1)

If it is EMPTY, then display "Queue is EMPTY!!! Deletion is not possible!!!" and terminate the function.

Step 2 - If it is NOT EMPTY, then display queue[front] as deleted element

Step 3-If front is equal to rear it means deleted element is the last value of the queue. The queue has become empty. In this case set front and rear both pointers to -1.

Step 4-If front is pointing to the last index value then set it to the first index value(circular array)

Step 5- else increase the value of front pointer by 1

Step 6-Exit

Algorithm for Display operation:

Step 1 - Check whether queue is EMPTY. (front ==rear== -1)

If it is EMPTY, then display "Queue is EMPTY!!!" and terminate the function.

Step 2 - If it is NOT EMPTY, then define an integer variable 'i' and set 'i = front'.

Step 3 - Check whether 'front <= rear', if it is TRUE, it means rear is left or equal to the front,then display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until 'i <= rear' becomes FALSE.

Step 4 - If 'front <= rear' is FALSE, then display 'queue[i]' value and increment 'i' value by one (i++). Repeat the same until'i <= SIZE - 1' becomes FALSE.

Step 6 - Set i to 0.

Step 7 - Again display 'cQueue[i]' value and increment i value by one (i++). Repeat the same until 'i <= rear' becomes FALSE.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Code:

1) Linear Queue:

```
#include <iostream>
using namespace std;

int queue[5];
int front=-1,rear=-1;

void enqueue(int x)
{
    if(rear==4)
    {
        cout<<"Overflow"<<endl;
        return;
    }
    if(front==-1 && rear==-1)
    {
        front=0;
        rear=0;
    }
    else
    {
        rear=rear+1;
    }
    queue[rear]=x;
}

void dequeue()
{
    if(front ==-1)
    {
        cout<<"Underflow"<<endl;
        return;
    }
    if(front==rear)
    {
        front=-1;
        rear=-1;
    }
    else
    {
        front=front+1;
    }
}
```

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

```
void display()
{
    if(front ==-1 && rear==-1)
    {
        cout<<"Underflow"<<endl;
        return;
    }
    for(int i=front ;i<rear+1;i++)
    {
        cout<<queue[i]<<" ";
    }
    cout<<endl;
}

int main()
{
    while(1)
    {
        cout<<"Enter: 1)Enqueue 2)Dequeue 3)Display "<<endl;
        int choose;
        cin>>choose;
        switch (choose )
        {
        case 1:
            cout<<"Enter element to be enqueued: ";
            int a;
            cin>>a;
            enqueue(a);
            break;
        case 2:
            cout<<"Dequeued element is: "<<queue[front]<<endl;
            dequeue();
            break;
        case 3:
            cout<<"Elements in queue are: ";
            display();
            break;
        default:
            cout<<"Enter valid input: ";
            break;
        }
    }
    return 0;
}
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

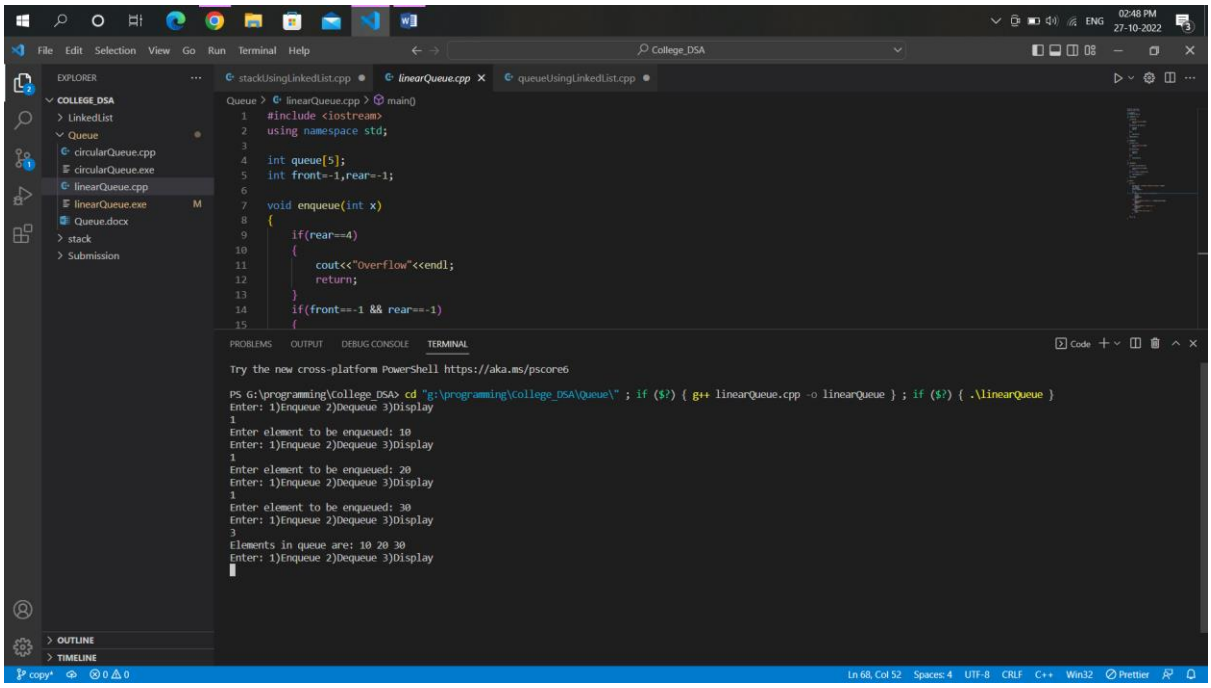
Department of Electronics & Telecommunication Engineering

Data Structures & Algorithms Lab (DJ19ECSBL1)

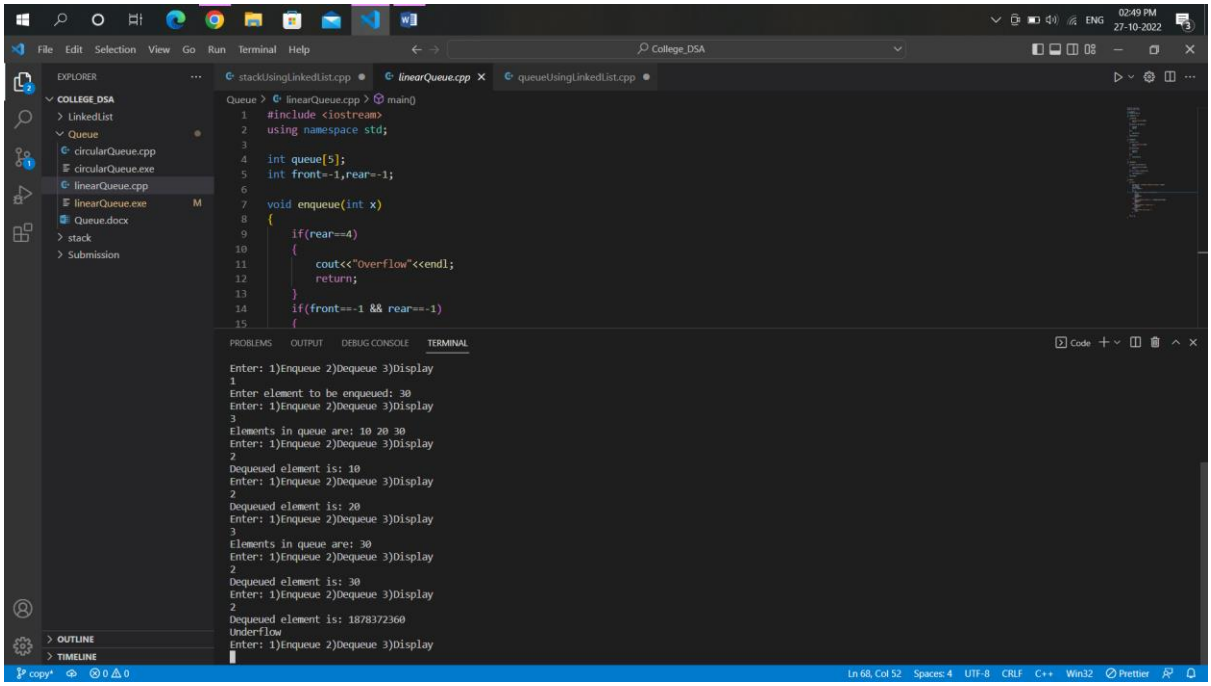
Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Output:

Enqueue and display:



Dequeue and display:

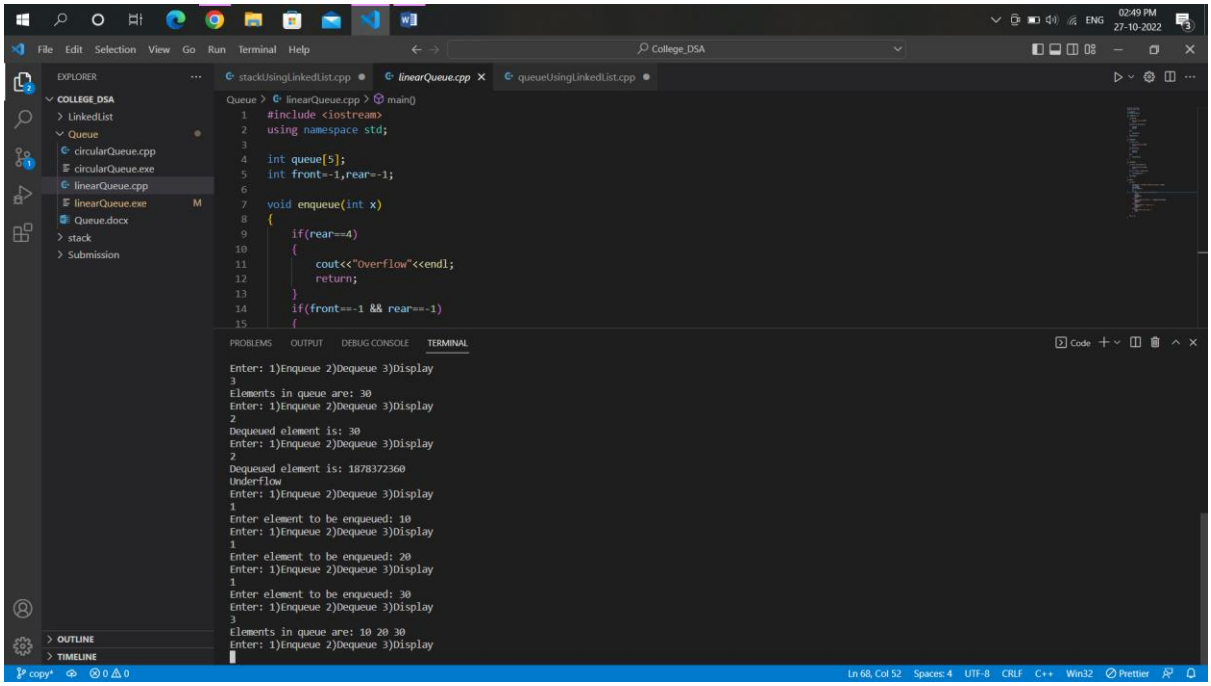


Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Display:



Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

2) Circular Queue:

Code:

Code:
```
#include <iostream>
using namespace std;

int queue[5];
int front=-1,rear=-1;

void enqueue(int x)
{
    if((front ==0 && rear== 4)|| (front==rear+1))
    {
        cout<<"Overflow"<<endl;
        return;
    }
    if(front==-1 && rear==-1)
    {
        front=0;
        rear=0;
    }
    else if(rear==4)
    {
        rear=0;
    }
    else
    {
        rear=rear+1;
    }
    queue[rear]=x;
}

void dequeue()
{
    if(front==-1 && rear==-1)
    {
        cout<<"Underflow"<<endl;
        return;
    }
    if(front==rear)
    {
        front=-1;
        rear=-1;
    }
    else if(front==4)
```

```
    {
        front=0;
    }
    else
    {
        front=front+1;
    }
}

void display()
{
    if(front==-1 && rear==-1)
    {
        cout<<"Underflow"<<endl;
        return;
    }
    if(front<rear)
    {
        for(int i=front;i<rear+1;i++)
        {
            cout<<queue[i]<<" ";
        }
        cout<<endl;
    }
    else
    {
        for(int i=front;i<5;i++)
        {
            cout<<queue[i]<<" ";
        }
        for(int i=0;i<rear+1;i++)
        {
            cout<<queue[i]<<" ";
        }
        cout<<endl;
    }

}

int main()
{
    while(1)
    {
        cout<<"Enter: 1)Enqueue 2)Dequeue 3)Display "<<endl;
        int choose;
```

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

```
        cin>>choose;
        switch (choose )
        {
        case 1:
            cout<<"Enter element to be enqueued: ";
            int a;
            cin>>a;
            enqueue(a);
            break;
        case 2:
            cout<<"Dequeued element is: "<<queue[front]<<endl;
            dequeue();
            break;
        case 3:
            cout<<"Elements in queue are: ";
            display();
            break;
        default:
            cout<<"Enter valid input: ";
            break;
        }
    }
    return 0;
}

// 10 20 30 40 50
// 20 30 40 50
//       30 40 50
// 60       40 50
// 60          50
// 60
//
```

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

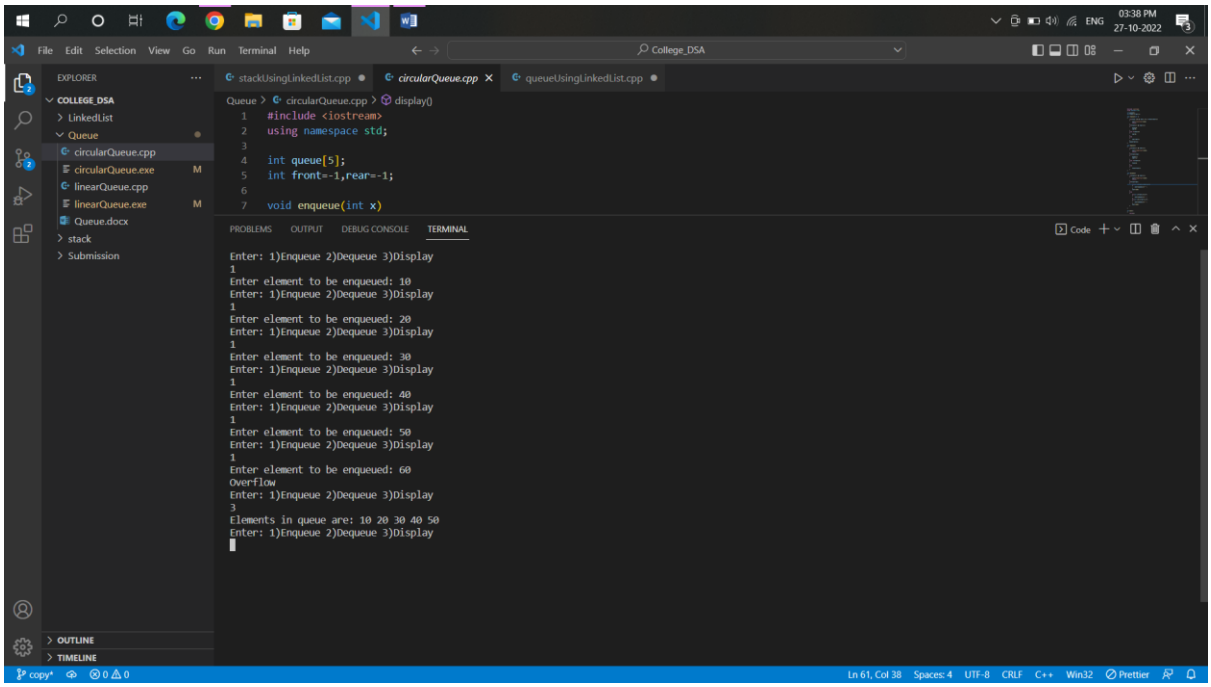
Department of Electronics & Telecommunication Engineering

Data Structures & Algorithms Lab (DJ19ECSBL1)

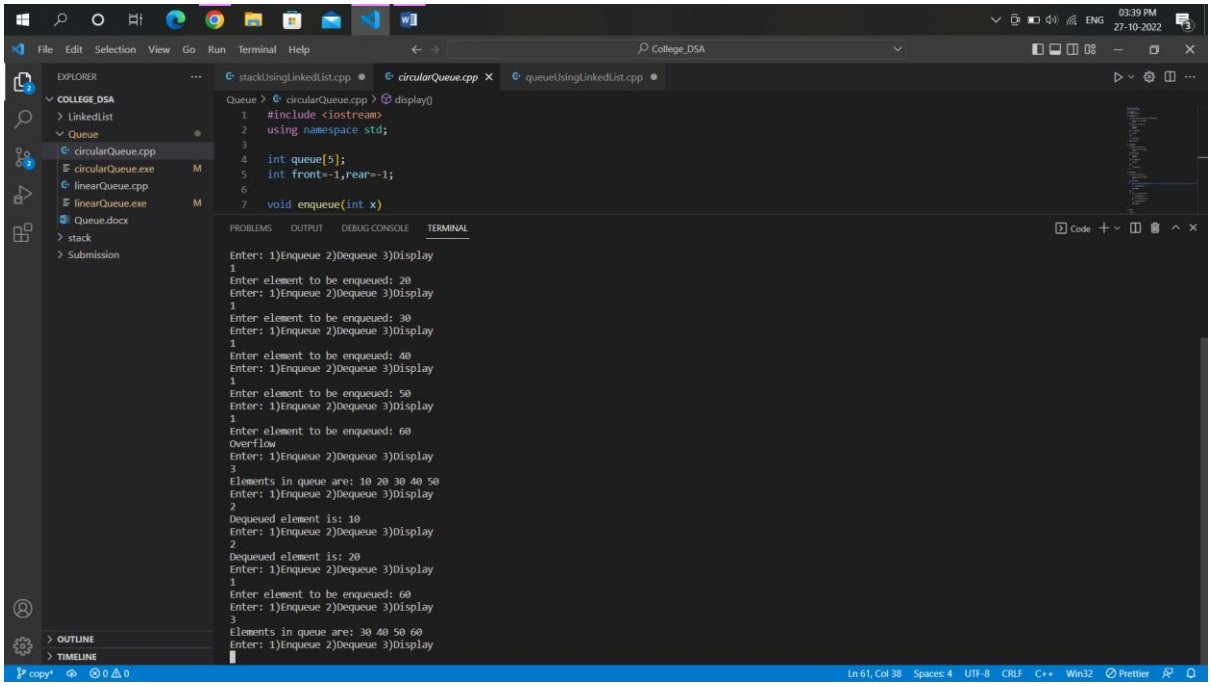
Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Output:

Enqueue and Display:



Dequeue and display:



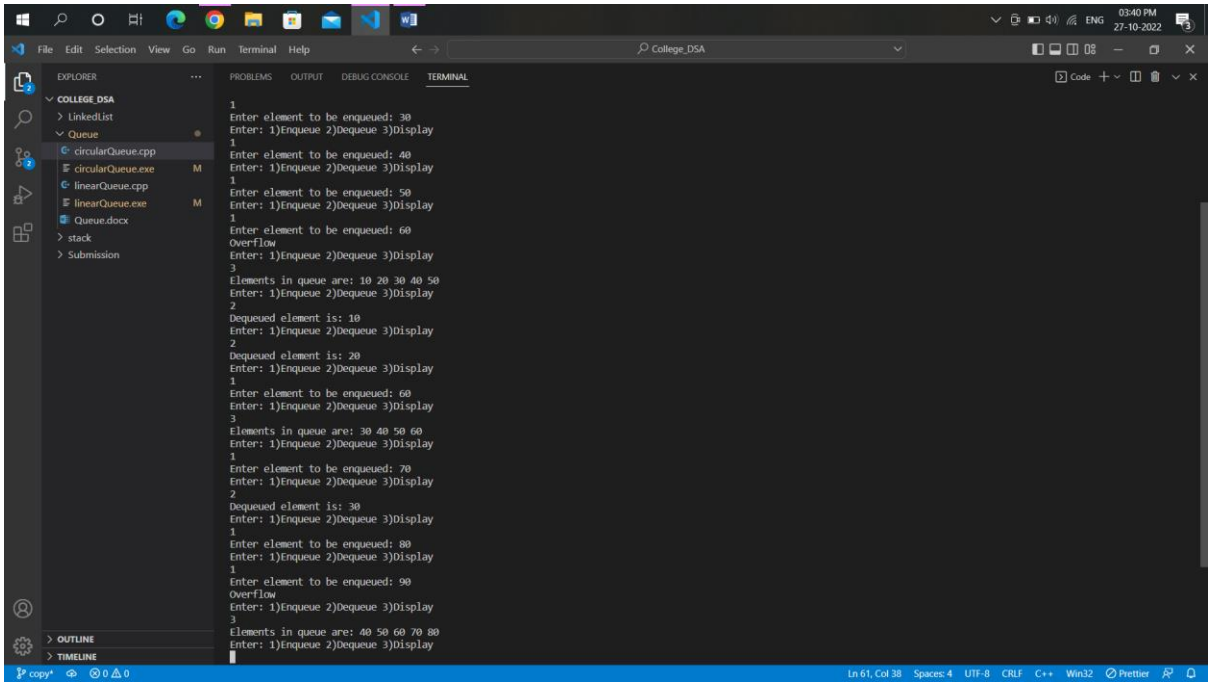
Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Electronics & Telecommunication Engineering

Data Structures & Algorithms Lab (DJ19ECSBL1)

Name : Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch:E1-4**

Implementation of circular queue:



Result and Conclusion:

1) **Implemented linear queue using array.**
2) **Operations enqueue, dequeue are done.**
3) **Linear array is not memory efficient as circular array.**
4) **Implemented circular queue using array.**
5) **Performed operations on circular queue and checked its memory efficiency.**