Page No.:
Date:

Experiment no. 5

→ Aim - A double-ended queue (deque) is a linear list in which additions & deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add & delete elements from either end of the deque.

Pre-requisite:

Knowledge of Queue
Types of Queue
Knowledge of double ended queue & different operations that can be performed on it.

Objective:
To simulate deque with function to add & delete elements from either end of the deque.

Input:
Size of array, elements in the queue :
Outcome:
Result of deque with functions to add & delete elements from either end of the deque.

Theory:

Double-Ended Queue

A double-ended queue is an abstract data type similar to an ~~similar~~ simple queue, it allows you to insert & delete from both sides means items can be added or deleted from the front or rear end.

Page No.:
Date:

Algorithm for Insertion at rear end

Step-1 : [Check for overflow] if (rear == MAX) Print ("Queue is Overflow"); return;
Step-2 [Insert element] else
rear = rear + 1;
q[rear] = no;
[Set rear and Front pointer]

IF rear = 0
rear ; If front = 0
Front ; Step-3 : return

Implementation of Insertion at rear end.

```
Void add_items_rear()
{
 int num;
printF ("\n Enter Items to insert : "); scanF("%d", &num); If (rear == MAX)
 {
  printF("\n Queue is OverFlow"); return;
}
  else
{
  rear++ ; q[rear] = num ; If (rear == 0) rear = 1 ; If (Front == 0) front = 1;
```

Algorithm for Insertion at front end

Step 1: [Check for the front position] if (front <= 1)
Print ("Cannot add item at front end"); return;
Step 2: [Insert at Front] else
front = front - 1 ; q[Front] = no ; Step-3 : Return

Page No.:
Date:

Implementation of Inseration at front end

```
Void add_item_front()
{
 int num;
printF("\n Enter item to insert:"); scanF("%d",&num);

iF(Front<=1)
{
printF("\n Cannot add item at front end");
return;
}
else
{
Front--; q[Front]=num;
}
}
```

Algorithm for Deletion from front end

Step-1 [Check for front pointer] if front=0
print("Queue is Underflow");
return;
Step-2 [Perform deletion] else
no=q[Front];
print("Deleted element is",no); [Set front and rear pointer]
If front=rear front=0; rear=0;
else front=front+1;
Step-3:
Return

Page No.:
Date:

Implementation of deletion from front end

```
Void delete_item_Front()
{
 int num; if (front == 0)
 {
 printF("\n Queue is Underflow \n");
 return;
 }
 else
 {
  num = q[Front];
printF ("\n Deleted items is %d \n", num); if (Front == rear)
 {
 Front = 0; rear = 0;
 }
 else
 {
 Front++;
 }
 }
 }
```

Algorithm for deletion from rear end
Step-1: [Check for the rear pointer]
 If rear = 0
print ("Cannot delete value at rear end");
 return;
Step 2: [Perform deletion] else
 no = q[rear];
[Check for the front and rear pointer] If Front = rear
 front = 0 ; rear = 0; else

```
rear = rear - 1;
print("Deleted element is", no);
```
Step 3 - Return

Implementation of Deletion from rear end

```
Void delete_item_rear()
{
printF("\n Cannot delete item at rear end\n");
return;
}
else
{
num = q[rear]; if(Front == rear)
{
Front = 0; rear = 0;
}
else
{
rear--;
printF("\n Deleted items is %d\n", num);
}
}
}
```

Conclusion:
By this way, we can perform operations on double ended queue.

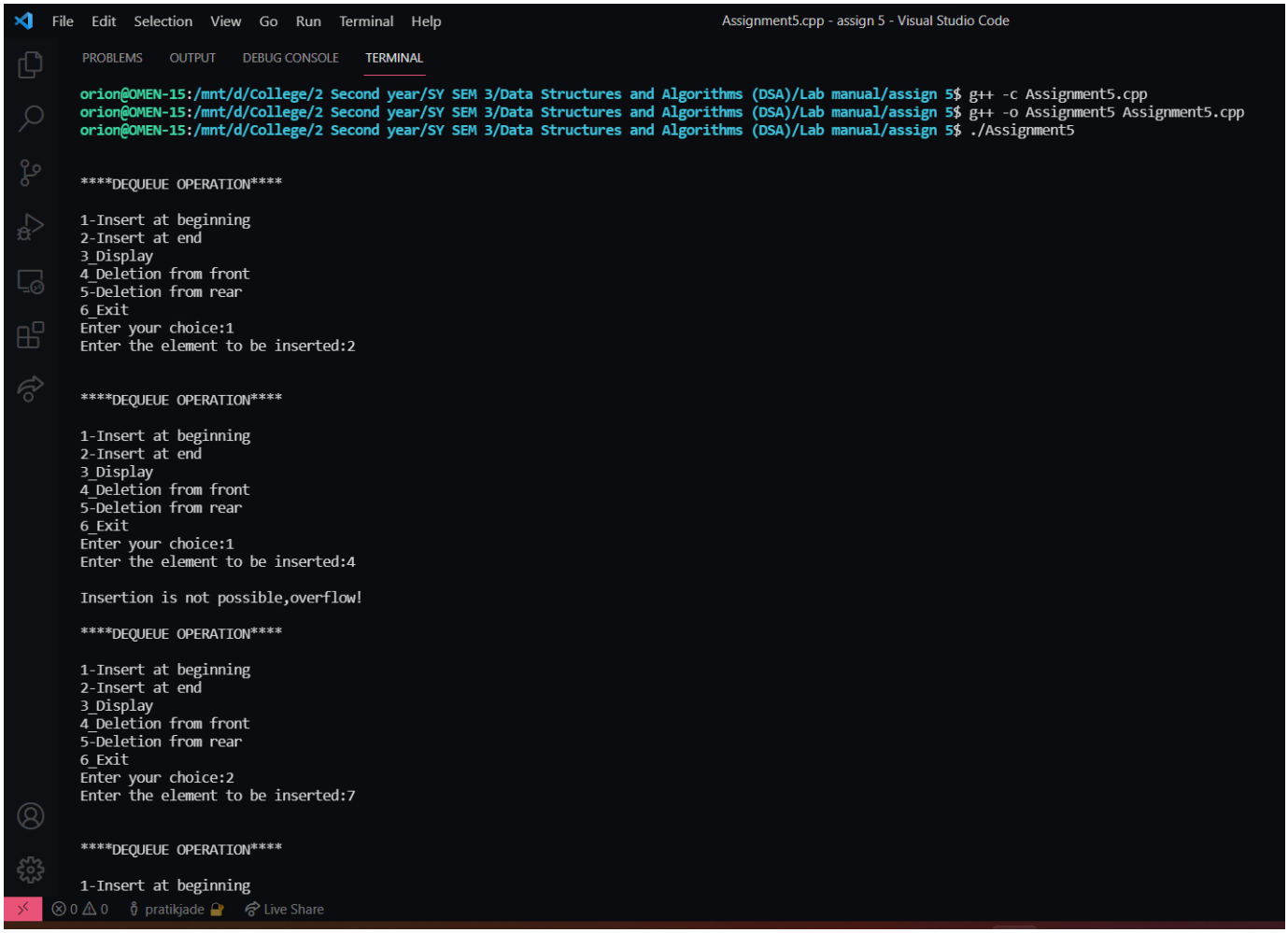
Program –

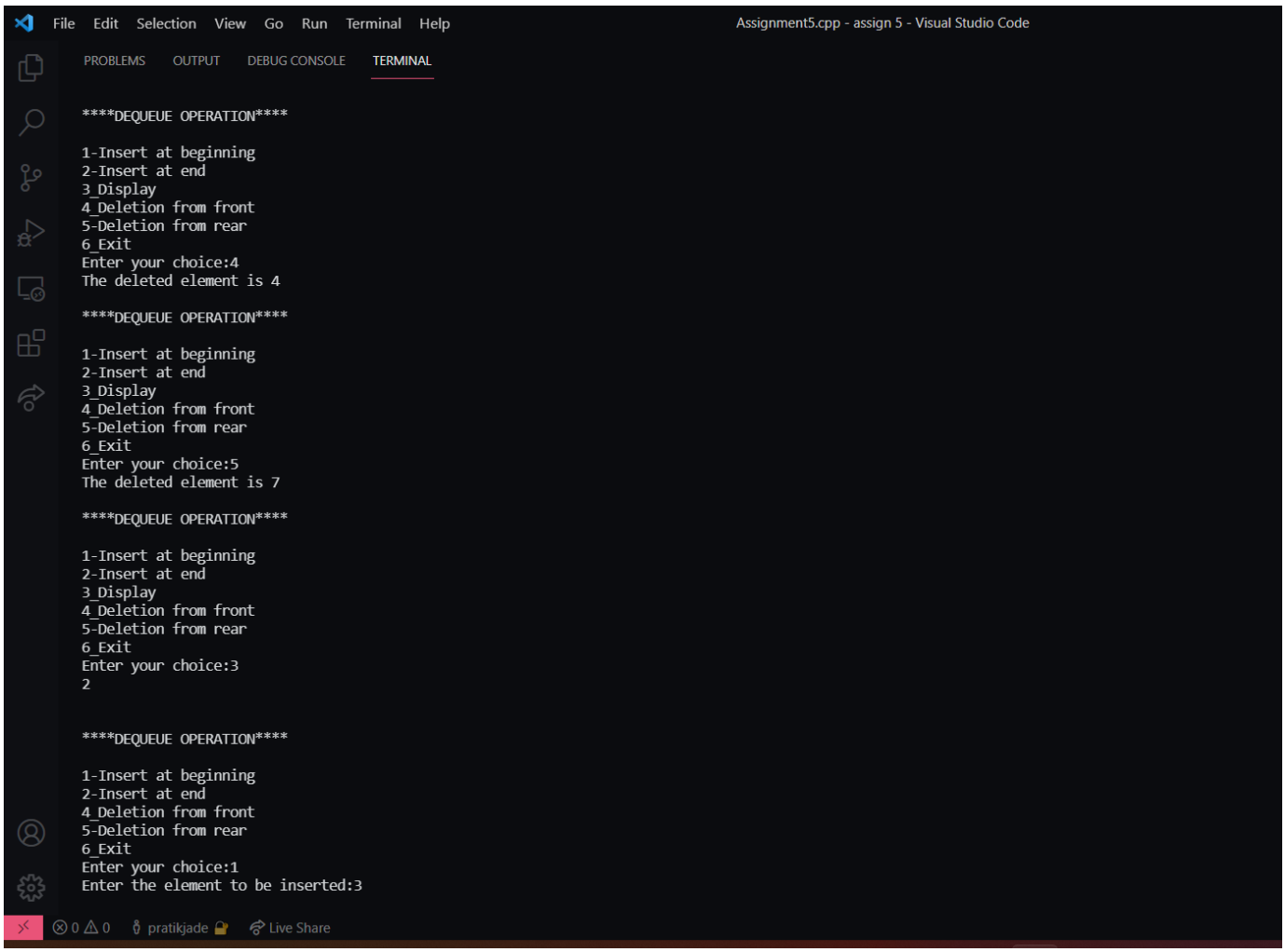
```
#include <iostream>
using namespace std;
#define SIZE 5
class dequeue
{
    int a[10], front, rear, count;

public:
    dequeue();
    void add_at_beg(int);
    void add_at_end(int);
    void delete_fr_front();
    void delete_fr_rear();
    void display();
};
dequeue::dequeue()
{
    front = -1;
    rear = -1;
    count = 0;
}
void dequeue::add_at_beg(int item)
{
    int i;
    if (front == -1)
    {
        front++;
        rear++;
        a[rear] = item;
        count++;
    }
    else if (rear >= SIZE - 1)
    {
        cout << "\nInsertion is not possible,overflow!";
    }
    else
    {
        cout << "\nInsertion is not possible,overflow!";
        for (i = count; i >= 0; i--)
        {
            a[i] = a[i - 1];
        }
        a[i] = item;
        count++;
        rear++;
    }
}
void dequeue::add_at_end(int item)
{
    if (front == -1)
    {
        front++;
        rear++;
```

```
        a[rear] = item;
        count++;
    }
    else if (rear >= SIZE - 1)
    {
        cout << "\nInsertion is not possible,overflow!";
        return;
    }
    else
    {
        a[++rear] = item;
    }
}
void dequeue::display()
{
    for (int i = front; i <= rear; i++)
    {
        cout << a[i] << "\n";
    }
}
void dequeue::delete_fr_front()
{
    if (front == -1)
    {
        cout << "Deletion is not possible: Dequeue is empty";
        return;
    }
    else
    {
        if (front == rear)
        {
            front = rear = -1;
            return;
        }
        cout << "The deleted element is " << a[front];
        front = front + 1;
    }
}
void dequeue::delete_fr_rear()
{
    if (front == -1)
    {
        cout << "Deletion is not possible: Dequeue is empty";
        return;
    }
    else
    {
        if (front == rear)
        {
            front = rear = -1;
        }
        cout << "The deleted element is " << a[rear];
        rear = rear - 1;
    }
}
```

```
int main()
{
    int c, item;
    dequeue d1;
    do
    {
        cout << "\n\n****DEQUEUE OPERATION****\n";
        cout << "\n1-Insert at beginning";
        cout << "\n2-Insert at end";
        cout << "\n3_Display";
        cout << "\n4_Deletion from front";
        cout << "\n5-Deletion from rear";
        cout << "\n6_Exit";
        cout << "\nEnter your choice:";
        cin >> c;
        switch (c)
        {
        case 1:
            cout << "Enter the element to be inserted:";
            cin >> item;
            d1.add_at_beg(item);
            break;
        case 2:
            cout << "Enter the element to be inserted:";
            cin >> item;
            d1.add_at_end(item);
            break;
        case 3:
            d1.display();
            break;
        case 4:
            d1.delete_fr_front();
            break;
        case 5:
            d1.delete_fr_rear();
            break;
        case 6:
            exit(1);
            break;
        default:
            cout << "Invalid choice";
            break;
        }
    } while (c != 7);
    return 0;
}
```

Output-


File Edit Selection View Go Run Terminal Help
Assignment5.cpp - assign 5 - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
orion@OMEN-15:/mnt/d/College/2 Second year/SY SEM 3/Data Structures and Algorithms (DSA)/Lab manual/assign 5$ g++ -c Assignment5.cpp
orion@OMEN-15:/mnt/d/College/2 Second year/SY SEM 3/Data Structures and Algorithms (DSA)/Lab manual/assign 5$ g++ -o Assignment5 Assignment5.cpp
orion@OMEN-15:/mnt/d/College/2 Second year/SY SEM 3/Data Structures and Algorithms (DSA)/Lab manual/assign 5$ ./Assignment5
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:1
Enter the element to be inserted:2
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:1
Enter the element to be inserted:4
Insertion is not possible,overflow!
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:2
Enter the element to be inserted:7
****DEQUEUE OPERATION****
1-Insert at beginning
0 0 pratikjade Live Share


File Edit Selection View Go Run Terminal Help
Assignment5.cpp - assign 5 - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:4
The deleted element is 4
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:5
The deleted element is 7
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
3_Display
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:3
2
****DEQUEUE OPERATION****
1-Insert at beginning
2-Insert at end
4_Deletion from front
5-Deletion from rear
6_Exit
Enter your choice:1
Enter the element to be inserted:3
0 0 pratikjade Live Share