DSA LAB 4

K036

Arjun Mehta

B. Tech CSE Cybersecurity


```
#include <iostream>
using namespace std;
#define MAX 10
class Queue
{
private:
  int que[MAX];
  int front, rear;
public:
  Queue() : front(-1), rear(-1) {}
  bool isEmpty()
  {
     return (front == -1 && rear == -1);
  }
  bool isFull()
  {
     return (rear == MAX - 1);
  }
  int size()
```

```
{
    if (isEmpty()) return 0;
    return (rear - front + 1);
}
void enqueue(int c)
{
    if (isFull())
        {
        cout << "Overflow\n";
        return;
    }
    if (isEmpty())
        {
        front = 0;
    }
    rear++;
    que[rear] = c;
}
int dequeue()
{
    if (isEmpty())
        {
        cout << "Underflow\n";
        return -9999;
    }
```

```
    int value = que[front];
    if (front == rear)
    {
        front = rear = -1;
    } else {
        front++;
    }
    return value;
}
int frontElement()
{
    if (isEmpty())
        {
        cout << "Queue is empty\n";
        return -9999;
    }
    return que[front];
}
void display() {
    if (isEmpty())
        {
        cout << "Queue is empty\n";
        return;
    }
    cout << "Queue is: ";
```

```
        for (int i = front; i <= rear; i++)
            {
            cout << que[i] << " ";
        }
        cout << endl;
    }
};
int main()
{
    Queue q;
    int choice;
    do
        {
        cout << "Enter 1: Enqueue Operation \n"
             << "2: Dequeue Operation \n"
             << "3: Return Front Element \n"
             << "4: Check if Queue is Empty \n"
             << "5: Size of Queue \n"
             << "6: Display Queue \n"
             << "0: Terminate Operations \n";
        cin >> choice;
        switch (choice)
        {
        case 1:
            int n;
```

```
    cout << "Enter the number to be inserted: ";
    cin >> n;
    q.enqueue(n);
    break;
case 2:
    cout << q.dequeue() << "\n";
    break;
case 3:
    cout << q.frontElement() << "\n";
    break;
case 4:
    if (q.isEmpty())
        cout << "The Queue is Empty\n";
    else
        cout << "The Queue is Not Empty\n";
    break;
case 5:
    cout << "The size of the queue is: " << q.size() << "\n";
    break;
case 6:
    q.display();
    break;
case 0:
    break;
default:
```

```
            cout << "Invalid Input\n";
        }
    } while (choice != 0);
    cout << "The operations are Terminated\n";
    return 0;
}
```

```
Enter 1: Enqueue Operation
2: Dequeue Operation
3: Return Front Element
4: Check if Queue is Empty
5: Size of Queue
6: Display Queue
0: Terminate Operations
1
Enter the number to be inserted: 1
Enter 1: Enqueue Operation
2: Dequeue Operation
3: Return Front Element
4: Check if Queue is Empty
5: Size of Queue
6: Display Queue
0: Terminate Operations
1
Enter the number to be inserted: 2
Enter 1: Enqueue Operation
2: Dequeue Operation
3: Return Front Element
4: Check if Queue is Empty
5: Size of Queue
6: Display Queue
0: Terminate Operations
2
1
Enter 1: Enqueue Operation
2: Dequeue Operation
3: Return Front Element
4: Check if Queue is Empty
5: Size of Queue
6: Display Queue
0: Terminate Operations
6
Queue is: 2
Enter 1: Enqueue Operation
2: Dequeue Operation
3: Return Front Element
4: Check if Queue is Empty
5: Size of Queue
6: Display Queue
0: Terminate Operations
5
The size of the queue is: 1
Enter 1: Enqueue Operation
2: Dequeue Operation
3: Return Front Element
4: Check if Queue is Empty
5: Size of Queue
6: Display Queue
0: Terminate Operations
```

Queue:

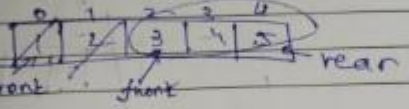
Queues

enqueue(int);

int dequeue();

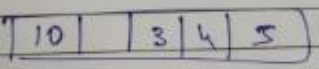
Types:

1) linear



rear = MAX-1

2) Circular Queue

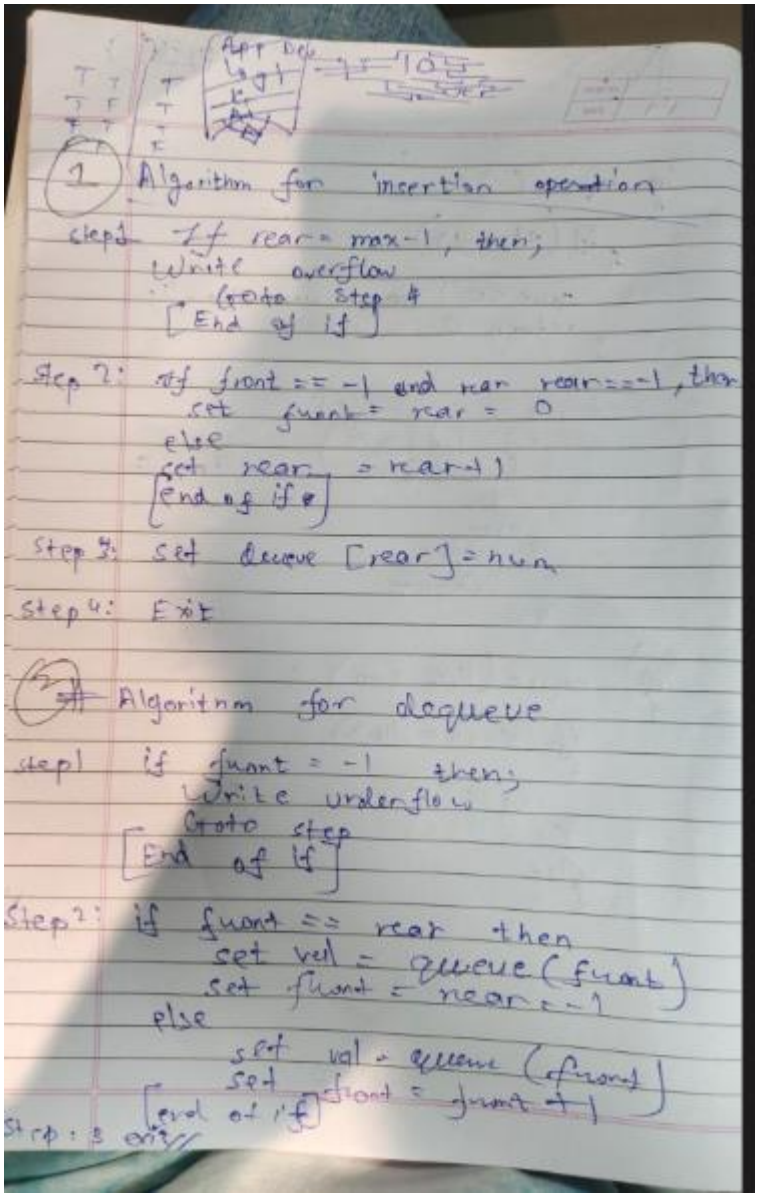


3) Double faced Queue

int que [MAX]

int f, r;

f = r = -1;



1. Algorithm for insertion operation

step1 If rear = max-1, then;
Write overflow
(goto step 4
[End of if]

Step 2: If front == -1 and rear rear==-1, then
set front = rear = 0
else
set rear = rear+1)
[end of if]

Step 3: set Queue [rear] = num

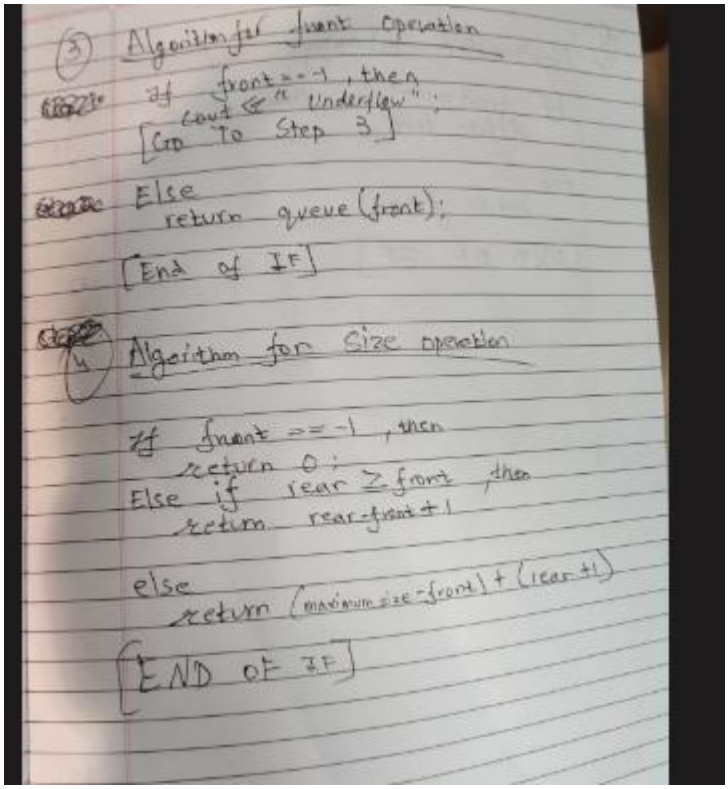
step 4: Exit

2. Algorithm for dequeue

step1 if front = -1 then;
Write underflow
Goto step
[End of if]

Step 2: if front == rear then
set val = queue (front)
set front = rear = -1
else
set val = queue (front)
Set front = front + 1
[end of if]

Step : 3 exit



(3) Algorithm for front operation

If front == -1, then
cout << "Underflow";
[Go To Step 3]

Else
return queue(front);

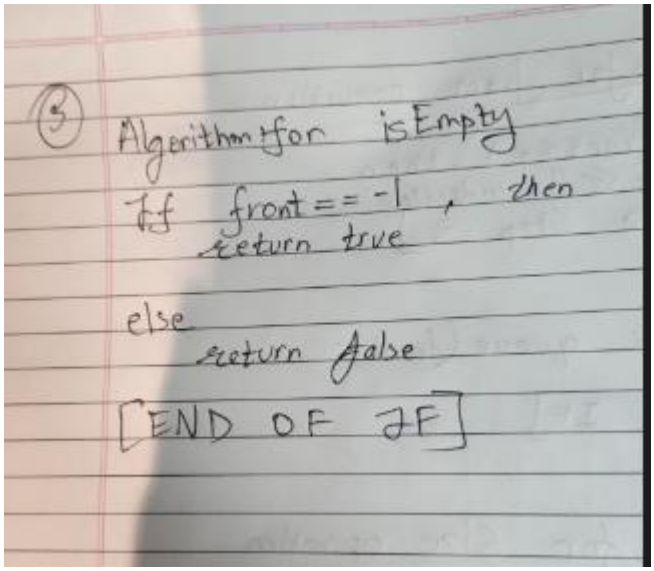
[End of IF]

(4) Algorithm for size operation

If front == -1, then
return 0;
Else if rear ≥ front, then
return rear-front+1

else
return (maximum size - front) + (rear+1)

[END OF IF]



(5) Algorithm for isEmpty

If front == -1, then
return true

else
return false

[END OF IF]