Queue

last in

| 4 | 3 | 2 | 1 |

Ticket Counter

first out

FIFO

1) Insert → Enqueue
2) Delete → Dequeue

STL

1) Array ✓ →
2) Dynamic Array ✓
3) Linked List ✓

→ insert at end() → enqueue
→ Deletion at begin() → Dequeue.

---

front = -1
rear = -1

```
Class Queue
{
  public:
    int q[100];
    int front;
    int rear;
    int size;
};
```
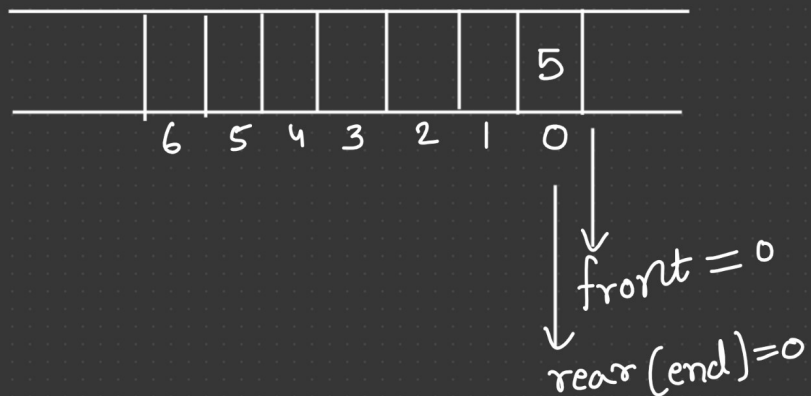
Queue qu;

1) Void Enqueue(int v)
   {
       if (rear == -1)
       {
           rear++; front++;
           q[rear] = v;
       }
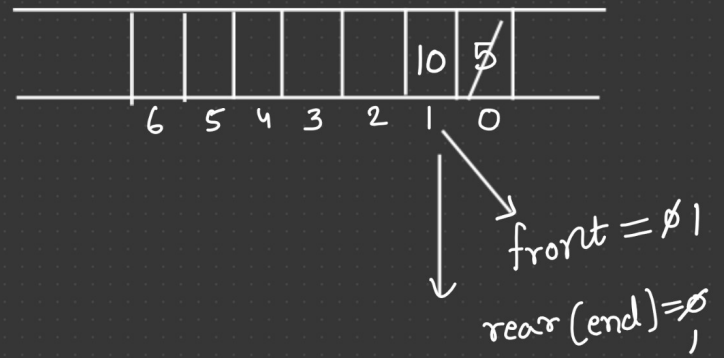   else if (rear == Size-1)
           cout << "Overflow";
```

| | | | | | | | | 5 | |

6  5  4  3  2  1  0

front = 0

rear (end) = 0

```
    else
    {
        rear++;
        q[rear] =v;
    }
}
```



```
               | | | | | |10| 5̶|
                6 5 4 3 2  1  0
```

front = 0̶ 1

rear (end) = 0̶ ;

```
2)  void  Dequeue()
    {
        if (front == -1)
            cout <<" underflow";
        else if ( rear == front)
        {
            cout<< q[front] <<" deleted";
            front = rear = -1;
        }
        else {
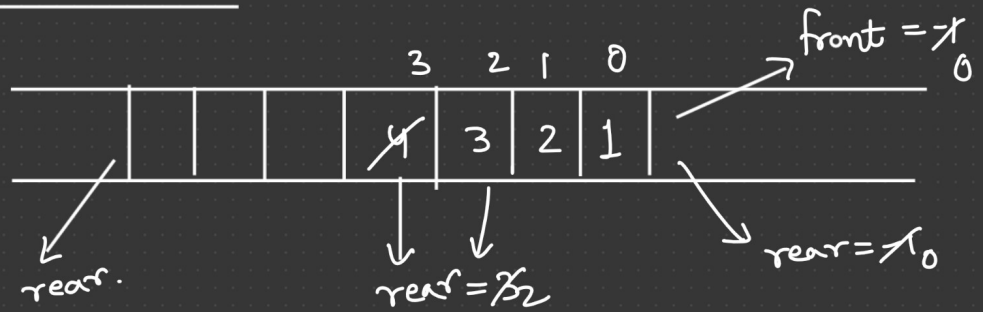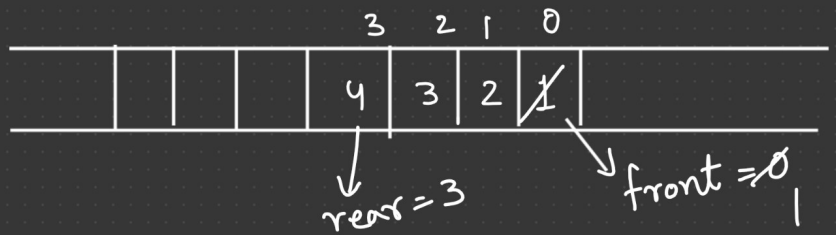            cout<< q[front] <<" deleted";
            front ++.
        }
    }
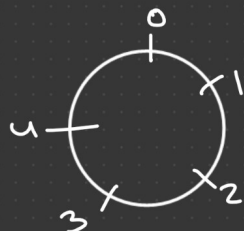
    void print()
    {
        for( i= front ; i <= rear ; i++)
            cout << q[i]<" ";
    }
```

Double Ended Queue :—

1) void insertAtEnd(int v)
```
{
    if( rear == -1)
    {
        front++ ; rear++;          ──────→ O(1)
        q[rear] =v;
    }
    else if ( rear == size-1)
    {                              ──────→ Drawback
        cout << "overflow";
    }
    else {
        q[++rear] = v;
    }
}
```

$0, 1, 2 \ldots \ldots \neq \textcircled{8}$

2) void deletionAtEnd( )
```
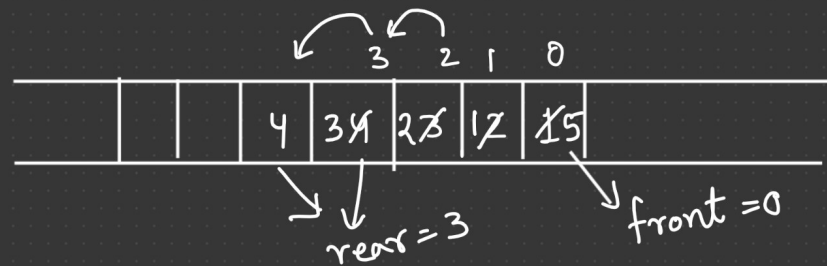{
    if (rear == -1)
    {
        cout << "underflow";       ──────→ O(1)
    }
    else if ( rear == front)
    {
        cout << q[rear] <" deleted";
        front = rear = -1;
    }
    else {
        cout << q[rear--] << "deleted";
    }
}
```

```
3) void deleteAtBegin()
   {
       if (front == -1)
          cout << "underflow";
       else if (rear == front)
       { cout << q[front] << " deleted";
```

$\longrightarrow O(1)$

```
          front = rear = -1;
       }
       else
       { cout << q[front++] << " deleted";
       }
   }
```



```
4) void insertAtBegin(int v)
   {
       if (front == -1)
       {
          front++; rear++;
          q[front] = v;
       }
       else if (rear == size-1)
          cout << "overflow";
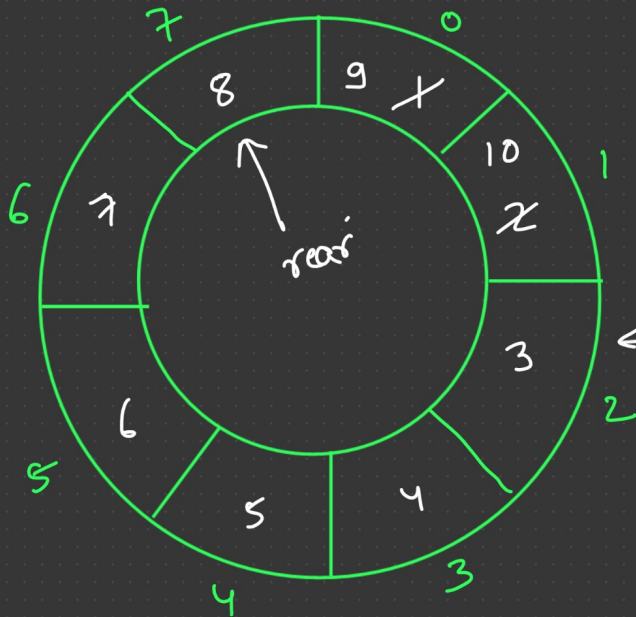       else {
          for (int i = rear; i >= front; i--)
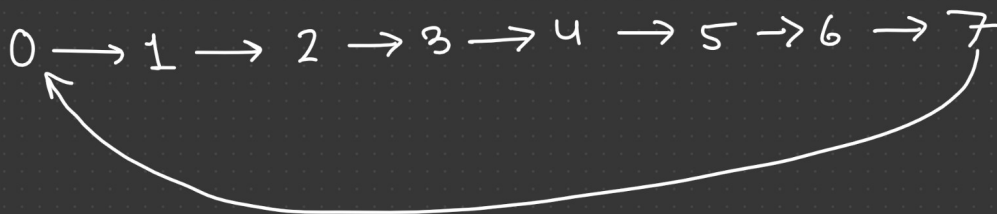          {
              q[i+1] = q[i];
          }
```

$\longrightarrow O(n)$

```
          rear++;
          q[front] = v;
       }
   }
```

Circular Queue :—

	4	3	2	1	0	
	5	~~4~~	~~3~~	~~2~~	~~1~~	

Problem ?

front = ~~0~~ ~~1~~ ~~2~~
4

rear = ~~1~~
~~2~~
~~3~~
4

7
8   9 ✗
          10
        ✗
6   7
     rear   front
         3
       6   2
5
      5   4
    4   3

while $(rear + 1 \ != front)$
&
insertion Allow
}

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$$

$0 \% 5 = 0$
$1 \% 5 = 1$
$2 \% 5 = 2$
$3 \% 5 = 3$
$4 \% 5 = 4$
$5 \% 5 = 0$
$6 \% 5 = 1$

Divide 5 → Remainder ⇒ $[0, 1, 2, 3, 4]$

0
4 —⊕— 1
3   2

$(rear + 1) \% 8 \implies [0 - 7]$

Concept

$$rear = (rear + 1) \% size$$