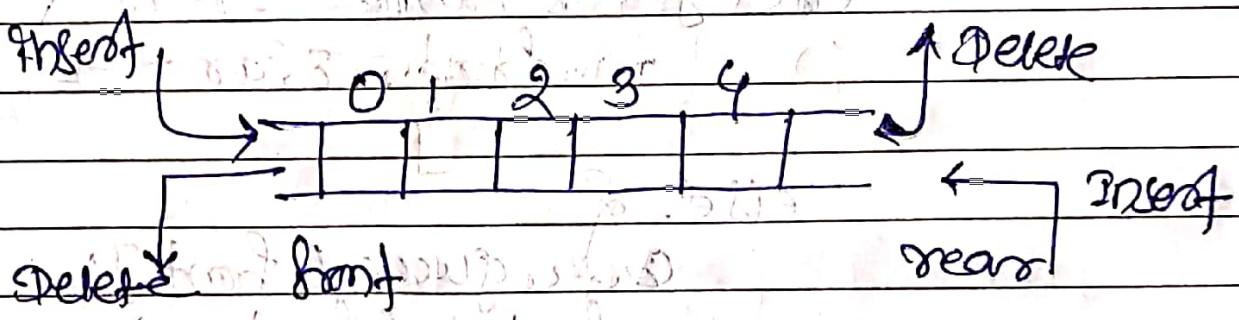
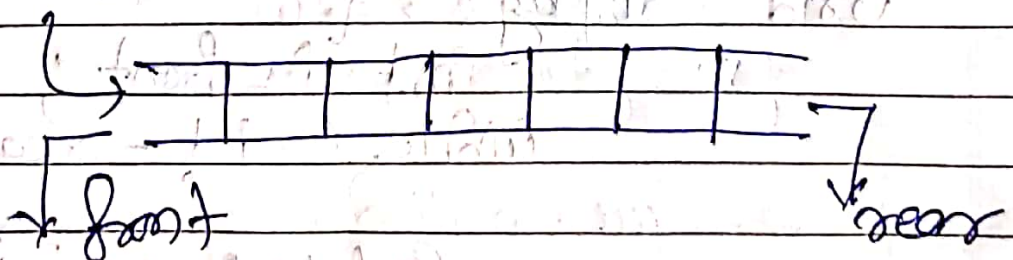
Dequeue (Double ended queue)

It is a linear list in which, enqueue and dequeue operation are allowed in both the end. It doesn't follow FIFO as well as LIFO rule.



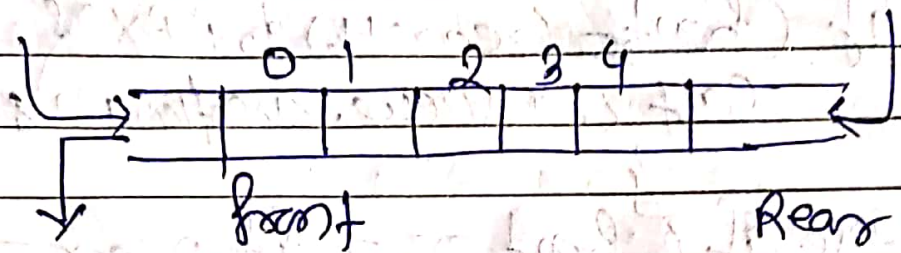
Types of Dequeue

Input restricted



Insertion is possible from only one end & deletion is possible from both end.

Output restricted



Deletion is possible from any one end & Insertion is possible from both end.

Application

(i) It is used in multiprocessor scheduling.
(ii) used in redo/undo operation.
(iii) used to check palindrome.

operation

(i) Enqueue front ( )
(ii) Enqueue rear ( )
(iii) Dequeue front ( )
(iv) Dequeue rear ( )
(v) getfront ( )
(vi) getlast ( )

```
void enqueue(int x){
if (front==(rear+1)%MAX){
    cout<<"queue overflow";
}
else if(front==-1 && rear==-1){
        front=rear=0;
        dequeue[front]=x;
}
else if(front==0){
        front=MAX-1;
        dequeue[front]=x;
}
else{
        front--;
        dequeue[front]=x;
}
}
```

```
void enqueRear (int x){
    if (front==(rear+1) % MAX){
        cout<<" queue overflow "<<endl;
    }
else if (front==-1 && rear==-1){
        front=rear=0;
        dequeue[rear]=x;
    }
else if (Rear==MAX-1){
        Rear=0;
        dequeue[rear]=x;
    }
else{
    Rear++;
    dequeue[rear]=x;
    }
}
```

```
void dequeuefront(){
  if(front==-1 && rear==-1){
    cout<<"queue underflow"<<endl;
  }
  else if(front==rear){
    front=rear=-1;
  }
  else if(front==MAX-1){
    cout<<dequeue[front];
    front=0;
  }
  else{
    cout<<dequeue[front];
    front++;
  }
}
```

```
void dequeueRear(){
  if(front==-1 && rear==-1){
    cout<<"queue underflow"<<endl;
  }
  else if(front==rear){
    front=rear=-1;
  }
  else if(rear==0){
    cout<<dequeue[rear];
    rear=MAX-1;
  }
  else{
    cout<<dequeue[rear];
    rear--;
  }
}
```