**LAB REPORT ON**

**Data Structures**



**Lab No. 2**

**Date 2020/09/18**

**Topic: Queue**

**Submitted By**  **Submitted To**

Tapendra Pandey

BCT-D Department of Electronics

075BCT093 and Computer Engineering

**INTRODUCTION**

**Queue:**

Queue is an ordered list in which insertions are done at one end called rear and deletions are done at other end called front. The first element inserted is the first one to be deleted. Hence, it is called First in First Out (FIFO) list.

A real-world example of queue can be a single-lane one-way road, where the vehicle enters first, exits first.

Basic queue operations involve initializing or defining the queue, utilizing it, and then completely erasing it from the memory. Some terms are:

1. Enqueue: It stores an item to the queue.
2. Dequeue: It removes an item from the queue.
3. Front/peek: It gets the element at the front of the queue without removing it.
4. Isfull: It checks if the queue is full.
5. Isempty: It checks if the queue is empty.

All these operations have time complexity O(1).

Stack is a “Last in First Out” structure whereas queue is “First in First Out structure”. In queue insertion and deletion happens from different side whereas in stack both operations happen from same side.

Deque:

Deque is a list in which the elements can be inserted or deleted at either end in constant time. It is also known as a head-tail list because elements can be added to or removed from either the front or the rear end.

**ALGORITHMS**

**1) Perform enqueue and dequeue operations in Linear Queue**

a) Enqueue operation

1: Initialize front=0 and rear=-1

2: define max array size N

3: initialize queue array of size N

4: read item

5: check if the array is full

If rear=N-1 return overflow

end if

6: rear = rear + 1

7: queue [rear] = item;

8: print queue

9: end

b) Dequeue operation

1: Repeat step 1 to 4 until front>=0

2: check if the queue is empty

If front<0 or front>rear

Return underflow

End if

3: Item=queue [front]

4: front=front + 1

5: print item

6: end

**2) Perform enqueue and dequeue operations in Circular Queue**

a) Enqueue operation

1: Initialize front=rear=-1

2: define max array size N

3: initialize queue [ ] array of size i = N

4: read item

5: check if the array is full

If ((front=0 and rear==N-1) or (rear= (front-1) %( N-1))) return overflow

End if

6: if front =-1

front=rear=0

queue [rear]=item

end if

7: else if

rear=N-1 and front! =0

rear=0

queue[rear]=item

end if

8: else

rear++

queue[rear]=item

9: print queue

10: end

b) Dequeue operation

1: check if the queue is empty

If front=-1 and front=-1

Return underflow

End if

2: Item=queue [front]

4: queue [front]=-1

5: if front =rear

front=rear=-1

End if

6: else if front=N-1

front=0

End if

7: else

Front++

8: print item

9: end

**3) Perform operations in Deque (Double ended queue) for:**

* 1. **Add at beginning**
  2. **Add at end**
  3. **Delete from beginning**
  4. **Delete from end**

a) Add at beginning

1: Initialize front=rear=-1

2: define max array size N

3: initialize queue [ ] array of size i = N

4: read item

5: check if the array is full

6: if (front=0 and rear=N -1 or front=rear+1)

return overload

End if

Else if(front = -1 and rear = -1)

front=rear=0

queue [front] = item

else if(front=0)

front=N-1

queue[front] = item

else

front--

queue[front] = item

7: print queue

8: end

\

b) Add at end

1: Initialize front=rear=-1

2: define max array size N

3: initialize queue [ ] array of size i = N

4: read item

5: check if the array is full

if(front=0 and rear=N -1 or front=rear+1)

return overload

else if(front = -1 and rear = -1)

front=rear=0

queue[rear] = item

else if(rear=N-1)

rear= 0

queue[rear] = item

else

rear++

queue[rear] = item

6: print queue

7: end

c) Delete from beginning

1: check if the queue is empty

2: if (front =-1 and rear= -1)

return underflow

3: else if (front= rear)

front=rear=-1

else if(front = N -1)

front =0

else

front++

4: print item

5: end

d) Delete from end

1: check if the queue is empty

if(front=-1 and rear=-1)

return underflow

end if

else if(front = rear)

front=rear = -1

else if(rear = 0)

rear = N-1

else

rear --

4: print item

5: end

**DISCUSSION AND CONCLUSION:**

In this lab, we learnt about the ordered list - queue and its operations. We implemented the knowledge, did enqueue, and dequeue operations in both linear data structures. Once if queue becomes full, we cannot insert the next element until all the elements are deleted from the queue. Therefore, to remove this limitation, we implemented circular queue and performed both enqueue and dequeue operations. In addition, we performed insertion and deletion of elements from both sides in double-ended queue.

Hence, all above algorithms were studied and implemented in lab.