In [ ]:
```
#Introduction of Queue
Queue is a linear data structure that will follow the principle of FIFO(First in first out)(The item
which will come first will be moved out first)
                                                                                                  )
```

In [ ]:
```
#Terminology/Operations used in Queue
1.Enqueue --> Insert an element in the queue(at the rear part)
2.Dequeue --> deletion of an element from the queue(from the front)
3.Front -->return the front value of the queue
len(stack)
```

In [ ]:
```
#Implementation of Queue
1.array
2.linkedlist
```

In [9]:
```
#Implementation of Queue using array
class Queue:
    def __init__(self):
        self.array=[]
        self.count=0
        self.front=0

    def enqueue(self,item):
        self.array.append(item)
        self.count=self.count+1
        return self.array

    def dequeue(self):
        if self.count==0:
            return "Queue is empty"

        element=self.array[self.front]
        self.front+=1
        self.count=self.count-1
        return element

    def Front(self):
        if self.count==0:
            return "Queue is empty"
        return self.array[self.front]


q=Queue()
print(q.enqueue(1))
print(q.enqueue(2))
print(q.enqueue(3))
q.enqueue(3)
print(q.Front())
print(q.dequeue())
print(q.Front())
print(q.dequeue())
print(q.Front())
```

```
[1]
[1, 2]
[1, 2, 3]
1
1
2
2
3
```

In [ ]:
```
#Implementation of QUEUE using linkedlist
1.create a node class
2.constructor of node class--->data
3.initilize next ==None


4.Create a Queue Class
5.Create
```

In [15]:
```
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None

class Queue:
    def __init__(self):
        self.head=None
        self.count=0
        self.tail=None

    def enqueue(self,data):
        newNode=Node(data)
        if self.head is None:
            self.head=newNode
            self.tail=newNode
        else:
            self.tail.next=newNode
        self.tail=newNode
        self.count+=1


    def dequeue(self):
        if self.head is None:
            return "Queue is empty"
        temp=self.head.data
        self.head=self.head.next
        self.count-=1
        return temp
    def Front(self):
        if self.head is None:
            return "Queue is empty"
        return self.head.data

q=Queue()
q.enqueue(1)
q.enqueue(2)
q.enqueue(3)
q.enqueue(3)
print(q.Front())
print(q.dequeue())
print(q.Front())
print(q.dequeue())
print(q.Front())
```

```
1
1
2
2
3
```

In [ ]:
```
#Applications of Queue:
real world:
    1.Ticket counter
    2.Metero train
    3.College Canteen
    4.college mess
Computer :
    1.Queue is used in Trees
    2.FcFS CPU scheduling
    3.Mail queue
    4.printer
```

In [ ]:
```
Variations of Quueue:
    1.Simple Queue
    2.Circular Quueue--> here rear is connected to front
    3.Priority Queue--> each and every element has its own prioerity and based on that insertion
    deletion will be done
    4.Dequeu -->Double Ended queue--> insertion and deletion from both end
```

In [ ]:
```
Variation of linkedlist:
    1.Singly Linkedlist
    2.Doubly linked list
    3.Circular linkedlist
    4.Doubly circular linkedlist
```

In [ ]:
```
#Implementation of Stack using linkedlist
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None

class stack:
    def __init__(self):
        self.head=None
        self.count=0

    def push(self,data):
        newdata=Node(data)
        newdata.next=self.head
        self.head=newdata
        self.count=self.count+1
    def pop(self):
        if self.count==0:
            print("empty stack")

        x=self.head.data
        self.head=self.head.next
        self.count=self.count-1
        return x
    def peek(self):
        if self.count==0:
            print("empty stack")
        return self.head.data

x=stack()
x.push(10)
x.push(20)
x.push(30)
x.push(40)
print(x.pop())
x.peek()
```