

In [20]:

```

1  # A
2  class Node:
3      def __init__(self,value):
4          self.value=value
5          self.next=None
6  class LinkedList:
7      def __init__(self):
8          self.head=None
9          self.tail=None
10     def insertAtFirst(self,value):
11         newnode=Node(value)
12         if self.head == None:
13             self.head=newnode
14             self.tail=newnode
15         else:
16             newnode.next=self.head
17             self.head=newnode
18     def insertAtEnd(self,value):
19         newnode=Node(value)
20         x=self.tail
21         if self.head == None:
22             self.head=newnode
23             self.tail=newnode
24         else:
25             self.tail.next=newnode
26             self.tail=newnode
27     def insertAtAfter(self,position,value):
28         newnode=Node(value)
29         if self.tail == None:
30             self.head=newnode
31             self.tail=newnode
32         else:
33             temp=self.head
34             while temp.value != position:
35                 temp=temp.next
36             a=temp.next
37             temp.next=newnode
38             newnode.next=a
39     def deleteAtFirst(self):
40         x=self.head
41         x=x.next
42         self.head=x
43     def deleteAtEnd(self):
44         p=self.head
45         q=p.next
46         while q.next != None:
47             p=p.next
48             q=q.next
49         p.next=None
50     def deleteAtAfter(self,position):
51         if self.tail == None:
52             pass
53         elif self.head.value==position:
54             self.deleteAtFirst()
55         else:
56             p=self.head
57             q=p.next
58             while q.value != position:
59                 p=p.next
60                 q=q.next
61             p.next=q.next
62     def Print(self):
63         x=self.head
64         print("Linked List:")
65         while x:
66             print(x.value,end=" ")
67             x=x.next
68 a=LinkedList()
69 a.insertAtFirst(5)
70 a.insertAtFirst(7)
71 a.insertAtEnd(2)
72 a.insertAtEnd(3)
73 a.deleteAtFirst()
74 a.deleteAtEnd()
75 a.insertAtAfter(5,8)
76 a.insertAtAfter(5,9)
77 a.deleteAtAfter(9)
78 a.insertAtAfter(2,4)
79 a.Print()

```

Linked List:
5 8 2 4

```
In [21]: 1 # B
2 class Node:
3     def __init__(self,value):
4         self.value=value
5         self.next=None
6 class ListStack:
7     def __init__(self,size):
8         self.size=size
9         self.top=0
10        self.head=None
11        self.tail=None
12    def isEmpty(self):
13        if self.top==0:
14            return True
15        else:
16            return False
17    def Push(self,value):
18        if self.top==self.size:
19            print("Stack Overflow !")
20        else:
21            self.top+=1
22            newnode=Node(value)
23            if self.head == None:
24                self.head=newnode
25                self.tail=newnode
26            else:
27                newnode.next=self.head
28                self.head=newnode
29    def Pop (self):
30        if self.isEmpty():
31            print("Stack Underflow !")
32        else:
33            self.top-=1
34            x=self.head
35            x=x.next
36            self.head=x
37    def Check(self):
38        if self.isEmpty:
39            True
40        else:
41            False
42    def Peek(self):
43        print("Peek value of stack is:",self.head.value)
44    def Count(self):
45        return "Number of elements in stack: "+str(self.top)
46    def Print(self):
47        x=self.head
48        print("\nTop to down.")
49        while x:
50            print("Stack: |_",x.value,"_|")
51            x=x.next
52 ob=ListStack(3)
53 ob.Push(7)
54 ob.Push(6)
55 ob.Push(5)
56 ob.Pop()
57 ob.Push(1)
58 ob.Peek()
59 print(ob.Count())
60 ob.Print()
```

Peek value of stack is: 1
Number of elements in stack: 3

Top to down.
Stack: |_ 1 _|
Stack: |_ 6 _|
Stack: |_ 7 _|

In [23]:

```

1  # C
2  class Node:
3      def __init__(self,value):
4          self.value=value
5          self.next=None
6  class ListQueue:
7      def __init__(self,size):
8          self.size=size
9          self.head=None
10         self.tail=None
11     def enqueue(self,value):
12         if self.Count()==self.size:
13             pass
14         else:
15             newnode=Node(value)
16             x=self.tail
17             if self.head == None:
18                 self.head=newnode
19                 self.tail=newnode
20             else:
21                 self.tail.next=newnode
22                 self.tail=newnode
23     def dequeue(self):
24         if self.head==None:
25             pass
26         else:
27             x=self.head
28             x=x.next
29             self.head=x
30     def isEmpty(self):
31         if self.Count==0:
32             return True
33         else:
34             return False
35     def Count(self):
36         x=self.head
37         count=0
38         while x:
39             count+=1
40             x=x.next
41         return count
42     def Printt(self):
43         x=self.head
44         print("Queue:")
45         while x:
46             print(x.value,end=" ")
47             x=x.next
48 ob=ListQueue(4)
49 ob.enqueue(1)
50 ob.enqueue(2)
51 ob.enqueue(3)
52 ob.enqueue(4)
53 ob.enqueue(5)
54 ob.enqueue(6)
55 ob.dequeue()
56 ob.Printt()
57 print("\nLength of queue is:",ob.Count())

```

Queue:

2 3 4

Length of queue is: 3

In [24]:

```

1  # HOME WORK (DOUBLE LINKED LIST)
2  class Node:
3      def __init__(self,value):
4          self.value=value
5          self.next=None
6          self.prev=None
7  class LinkedList:
8      def __init__(self):
9          self.head=None
10         self.tail=None
11     def insertAtFirst(self,value):
12         newnode=Node(value)
13         if self.head == None:
14             self.head=newnode
15             self.tail=newnode
16         else:
17             newnode.next=self.head
18             self.head.prev=newnode
19             self.head=newnode
20     def insertAtEnd(self,value):
21         newnode=Node(value)
22         x=self.tail
23         if self.head == None:
24             self.head=newnode
25             self.tail=newnode
26         else:
27             self.tail.next=newnode
28             newnode.prev=self.tail
29             self.tail=newnode
30     def insertAtAfter(self,position,value):
31         newnode=Node(value)
32         temp=self.head
33         while temp.value != position:
34             temp=temp.next
35         if self.tail == None:
36             self.head=newnode
37             self.tail=newnode
38         #elif temp.next==None:
39
40         elif self.head.next==None or temp.next==None:
41             self.insertAtEnd(value)
42         else:
43             temp=self.head
44             while temp.value != position:
45                 temp=temp.next
46             a=temp.next
47             temp.next=newnode
48             newnode.prev=temp
49             newnode.next=a
50             a.prev=newnode
51     def deleteAtFirst(self):
52         x=self.head
53         x=x.next
54         self.head=x
55         self.head.prev=None
56     def deleteAtEnd(self):
57         self.tail=self.tail.prev
58         self.tail.next=None
59     def deleteAtAfter(self,position):
60         if self.tail == None:
61             pass
62         elif self.head.value==position:
63             self.deleteAtFirst()
64         else:
65             p=self.head
66             q=p.next
67             while q.value != position:
68                 p=p.next
69                 q=q.next
70             q.next.prev=p
71             p.next=q.next
72     def Print(self):
73         x=self.head
74         print("Linked List:")
75         while x:
76             print(x.value,end=" ")
77             x=x.next
78     def PrintRev(self):
79         z=self.tail
80         print("\nReverse list using previous node.")

```

```
81         while z:
82             print(z.value,end=" ")
83             z=z.prev
84 a=LinkedList()
85 a.insertAtFirst(5)
86 a.insertAtFirst(7)
87 a.insertAtEnd(2)
88 a.insertAtEnd(3)
89 a.deleteAtFirst()
90 a.deleteAtEnd()
91 a.insertAtAfter(5,8)
92 a.insertAtAfter(5,9)
93 a.deleteAtAfter(9)
94 a.insertAtAfter(2,4)
95 a.Print()
96 a.PrintRev()
```

Linked List:

5 8 2 4

Reverse list using previous node.

4 2 8 5