1. Complete the following methods on Searching and Sorting.
2. You may use any language to complete the tasks.
3. You need to submit one single file containing all the methods/functions. You will get **10 days** to complete your lab. **NO LATE SUBMISSIONS WILL BE TAKEN**
4. The submission format MUST be maintained. You need to copy paste all your codes in ONE SINGLE .txt file and upload that. If format is not maintained, whole lab submission will be canceled.
5. If you are using JAVA, you must include the **Tester class** containing the main method which should test your other methods.
6. If you are using PYTHON, make sure your code has the methods invoked through **test statements.**
7. Usage of built in methods/libraries are NOT ALLOWED
8. The google form link for this lab is provided in BUX under LAB 6 Searching and Sorting subsection under the FALL21 CSE220 lab tab.

Searching and Sorting Lab

1. Sort an array **RECURSIVELY** using **selection** sort algorithm.

2. Sort an array **RECURSIVELY** using **insertion** sort algorithm.

3. Sort a **singly linked** sequential list using **bubble** sort algorithm.

4. Sort a **singly linked** sequential list using **selection** sort algorithm.

5. Sort a **DOUBLY linked** sequential list using **insertion** sort algorithm.

6. Implement **binary search** algorithm **RECURSIVELY**.

7. Implement a recursive algorithm to find the n-th Fibonacci number using memoization.

Assignment 06

CSE220

In [1]:

```
###Task 1

def Recursive_Selection_sort(array,x):
    index=len(array)-x-1
    max=array[index]
    max_idx=index

    if index >= 0:
        for i in range(0,index):
            if array[i] > max:
                max_idx = i
                max = array[i]
        temp=array[max_idx]
        array[max_idx]=array[index]
        array[index]=temp
        return Recursive_Selection_sort(array,x+1)

s=[20,30,40,60,50,10]
Recursive_Selection_sort(s,0)
print(s)
```

```
[10, 20, 30, 40, 50, 60]
```

In [2]:

```
###Task 2

def Recursive_insertion_sort(array,x):
    if len(array)<=x:
        return
    else:
        for i in range(x,0,-1):
            if array[i-1] > array[i]:
                temp=array[i-1]
                array[i-1]=array[i]
                array[i]=temp

    return Recursive_insertion_sort(array,x+1)

s=[20,30,40,60,50,10]
Recursive_insertion_sort(s,0)
print(s)
```

```
[10, 20, 30, 40, 50, 60]
```

In [3]:

```
###Task 3

class Node:
    def __init__(self, v, n):
        self.value=v
        self.next = n

class MyList:
    def __init__(self,a=None):
        self.head=None
        self.tail=None
        for i in a:
            newNode=Node(i,None)
            if self.head==None:
                self.head=newNode
                self.tail=newNode
            else:
                self.tail.next=newNode
                self.tail= newNode

    def showList(self):
        if self.head==None:
            print("Empty list")
            return
        n=self.head
        while n is not None:
            print(n.value)
            n=n.next
def bubble_sort(head):
    length=0
    n=head
    while n is not None:
        length=length + 1
        n = n.next
    for i in range(length-1,-1,-1):
        n = head
        for j in range(0,i):
            if n.value > n.next.value:
                temp = n.value
                n.value = n.next.value
                n.next.value = temp
            n=n.next

a=[10,20,40,12,3,1,9]
l1=MyList(a)
bubble_sort(l1.head)
l1.showList()
```

```
1
3
9
10
12
20
40
```

In [4]:

```
###Task 4

class Node:
    def __init__(self, v, n):
        self.value=v
        self.next = n

class MyList:
    def __init__(self,a=None):
        self.head=None
        self.tail=None
        for i in a:
            newNode=Node(i,None)
            if self.head==None:
                self.head=newNode
                self.tail=newNode
            else:
                self.tail.next=newNode
                self.tail= newNode

    def showList(self):
        if self.head==None:
            print("Empty list")
            return
        n=self.head
        while n is not None:
            print(n.value)
            n=n.next

def selelction_sort(head):
    if head.next is not None:
        n_one=head
        n_two=head.next
        while n_two is not None:
            if n_one.value > n_two.value:
                n_one = n_two
            n_two = n_two.next
        temp = head.value
        head.value = n_one.value
        n_one.value = temp
        selelction_sort(head.next)
    else:
        return

a=[10,20,40,12,3,1,9]
l1=MyList(a)
selelction_sort(l1.head)
l1.showList()
```

```
1
3
9
10
12
20
40
```

In [5]:

```
###Task 5

class Node:
    def __init__(self, v, n, p):
        self.value=v
        self.next = n
        self.prev=p
class Doubly_LinkedList:
    def __init__(self,a=None):
        self.head=None
        self.tail=None
        for i in a:
            newNode=Node(i,None,None)
            if self.head==None:
                self.head=newNode
                self.tail=newNode
            else:
                temp=self.tail
                self.tail.next=newNode
                self.tail= newNode
                newNode.prev=temp
    def showList(self):
        if self.head==None:
            print("Empty list")
            return
        n=self.head
        while n is not None:
            print(n.value)
            n=n.next

def insertion_sort(head):
    if head is not None:
        n=head
        while n.prev is not None:
            if n.prev.value > n.value:
                temp = n.prev.value
                n.prev.value = n.value
                n.value = temp
            else:
                break
            n=n.prev
        insertion_sort(head.next)

a=[40,60,20,10,30,50]
l=Doubly_LinkedList(a)
insertion_sort(l.head)
l.showList()
```

```
10
20
30
40
50
60
```

In [6]:

```
###Task 6

def binary_recursive_search(array,val,L,R):
    if L < R:
        index = (L+R)//2
        if val == array[index]:
            return index
        elif val > array[index]:
            L=index+1
        else:
            R=index-1
        return binary_recursive_search(array,val,L,R)
    else:
        return "Value Not Found"

array=[10,20,30,40,50,60]
print(binary_recursive_search(array,50,0,len(array)-1))
```

```
4
```

In [8]:

```
###Task 7

def fibonacci_memoi(n):
    array=[0]*999
    if n == 0:
        array[n]=0
    elif n==1:
        array[n]=1
    elif array[n] != 0:
        return array[n]
    else:
        array[n] = fibonacci_memoi(n-1) + fibonacci_memoi(n-2)
    return array[n]

print(fibonacci_memoi(7))
```

```
13
```

In [ ]:

```

```