#1 selection sort - recursive

def FindminIndex( a,start,end ):

if start == end:

return start

p = FindminIndex(a, start + 1, end)

if a[p]<a[start]:

return p

else:

return start

def recurSelectionSort(a, start, end):

if start == end:

return

k = FindminIndex(a, start, end)

if a[k]<a[start]:

a[k], a[start] = a[start], a[k]

recurSelectionSort(a, start+1, end)

a=[9,13,7,23,4,10]

recurSelectionSort(a, 0, len(a)-1)

print(a)

#2 insertion sort- recursive

def postion(arr, idx, count):

if arr[count] >= arr[idx]:

return count

return postion(arr, idx, count+1)

def shift(arr, exp, idx):

if idx==exp-1:

return

arr[idx+1]=arr[idx]

shift(arr,exp,idx-1)

def insertionSortRecursive(arr,idx):

if idx>=len(arr):

return

temp=arr[idx]

pos=postion(arr, idx, 0)

shift(arr,pos,idx-1)

arr[pos]=temp

insertionSortRecursive(arr,idx+1)

arr = [1,2,55,7,0,8,4]

n = len(arr)

insertionSortRecursive(arr,0)

print(arr)

#3 Singly linked list - Recursion - selection sort

class Node:

def \_\_init\_\_(self,elem,nxt):

self.elem=elem

self.next=nxt

class LinkedList:

def \_\_init\_\_(self,a):

self.head=Node(a[0],None)

tail=self.head

for i in range(1,len(a)):

n=Node(a[i],None)

tail.next=n

tail=tail.next

def bubble\_sort(self):

x=self.head

tail=None

while x!= None:

tail = x.next

while tail != None:

if tail.elem < x.elem:

temp = x.elem

x.elem = tail.elem

tail.elem = temp

tail = tail.next

x = x.next

a=[7,8,3,1,4,2]

j=LinkedList(a)

head=j.head

j.bubble\_sort()

def printforward(head):

while head!=None:

print(head.elem,end=" ")

head=head.next

printforward(head)

#4 Singly linked list - Recursion - selection sort

class Node:

def \_\_init\_\_(self,elem,nxt):

self.elem=elem

self.next=nxt

class LinkedList:

def \_\_init\_\_(self,a):

self.head=Node(a[0],None)

tail=self.head

for i in range(1,len(a)):

n=Node(a[i],None)

tail.next=n

tail=tail.next

a=[7,8,3,1,4,2]

j=LinkedList(a)

head=j.head

#check singly list

def printforward(head):

while head!=None:

print(head.elem,end=" ")

head=head.next

#Recursive selection sorting

def FindminIndexLinked(head):

if head.next==None:

return head

p = FindminIndexLinked(head.next)

if p.elem<head.elem:

return p

else:

return head

def recurSelectionSortLinked(head):

if head.next==None:

return

k = FindminIndexLinked(head)

if k.elem<head.elem:

temp=head.elem

head.elem=k.elem

k.elem=temp

recurSelectionSortLinked(head.next)

recurSelectionSortLinked(head)

printforward(head)

#5 Doubly linked list - Insertion sort

class Node:

def \_\_init\_\_(self, e, n, p):

self.val = e

self.next = n

self.prev = p

class DoublyLinkList:

def \_\_init\_\_(self, a):

self.head = Node(a[0],None,None)

tail = self.head

for i in range(1,len(a)):

new\_node = Node(a[i], None, None)

tail.next = new\_node

new\_node.prev = tail

tail= tail.next

def insertion\_sort(head1):

head = head1

tail = head1

while tail.next != None:

i = tail.next

while i != head:

if i.val < i.prev.val:

i.val,i.prev.val = i.prev.val, i.val

i = i.prev

tail = tail.next

return head

arr = [7,8,6,5,3,1]

j = DoublyLinkList(arr)

head1 = j.head

k=insertion\_sort(head1)

while(k!=None):

if k.next != None:

print(k.val, end=" ")

k=k.next

#6 binary search

def search(a, s, start,end):

if start>end:

return -1

mid=(start+end)//2

if a[mid]==s:

return mid

elif a[mid] >s:

return search(a,s,start,mid-1)

elif a[mid]<s:

return search(a,s,mid+1,end)

a=[10,20,30,40,50,60,70,80]

print(search(a, 60, 0,len(a)-1))

#7 fibonacci memoization

def fibo(n,a):

if n<=1:

return n

if a[n]!=None:

return a[n]

else:

a[n]=fibo(n-1,a)+fibo(n-2,a)

return a[n]

n=5

a=[None]\*(n+1)

print(fibo(n,a))