#Task 01

def selection\_sort(a, i=0):

if i >= len(a)-1:

return

index\_min = i

next = i+1

while next < len(a):

if a[next] < a[index\_min]:

index\_min = next

next += 1

temp = a[i]

a[i] = a[index\_min]

a[index\_min] = temp

selection\_sort(a, i+1)

return a

a = [2, 3, 1, 5, 8, 15, 20, 16]

print(selection\_sort(a))

#Task 2

def Insert\_sort(arr, n):

if n <= 1:

return

Insert\_sort(arr, n-1)

item = arr[n-1]

j = n-2

while (j >= 0 and arr[j] > item):

arr[j+1] = arr[j]

j = j-1

arr[j+1] = item

print(arr)

arr = [12, 11, 13, 5, 6]

Insert\_sort(arr, len(arr))

#Task 03- Sort a singly linked sequential list using bubble sort algorithm

class Node:

def \_\_init\_\_(self, value, next):

self.value = value

self.next = next

class MyList:

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

self.head = None

tail = None

self.size = len(a)

for i in a:

if a == None:

raise Exception("Invalid index")

else:

n = Node(i, None)

if self.head is None:

self.head = n

tail = n

else:

tail.next = n

tail = n

def bubble\_sort(self):

while self.head != None:

current = self.head

next = self.head.next

while next != None:

if current.value > next.value:

temp = current.value

current.value = next.value

next.value = temp

next = next.next

print(self.head.value, end=" ")

self.head = self.head.next

def show(self):

if self.head != None:

while self.head != None:

print(self.head.value, end=" ")

self.head = self.head.next

else:

print("Empty")

#Task 04

def selection\_sort(self):

i = self.head

while i != None:

index\_min = i

j = i.next

while j != None:

if index\_min.value > j.value:

index\_min = j

j = j.next

if index\_min.value != i.value:

temp = i.value

i.value = index\_min.value

index\_min.value = temp

print(i.value, end=" ")

i = i.next

a = [2, 3, 1, 5, 10, 9, 8, 15, 20, 16]

b = MyList(a)

# b.show()

# b.bubble\_sort()

b.selection\_sort()

#Task 05

class Node:

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

self.elements = elements

self.next = None

self.previous = None

class DoublyList:

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

self.head = None

self.tail = None

def addNode(self, elements):

newNode = Node(elements)

if(self.head == None):

self.head = self.tail = newNode

self.head.previous = None

self.tail.next = None

else:

self.tail.next = newNode

newNode.previous = self.tail

self.tail = newNode

self.tail.next = None

def sortList(self):

if(self.head == None):

return

else:

current = self.head

while(current.next != None):

index = current.next

while(index != None):

if(current.elements > index.elements):

temp = current.elements

current.elements = index.elements

index.elements = temp

index = index.next

current = current.next

def showResults(self):

current = self.head

if(self.head == None):

print("List is empty")

return

while(current != None):

print(current.elements, end="<-->"),

current = current.next

print()

a = DoublyList()

a.addNode(7)

a.addNode(10)

a.addNode(4)

a.addNode(15)

a.addNode(2)

print("Original list: ")

a.showResults()

a.sortList()

print("Sorted List")

a.showResults()

#Task 6

def binary\_search(arr, left, right\_side, key):

if right\_side >= left:

mid = (right\_side + left) // 2

if arr[mid] == key:

return mid

elif arr[mid] > key:

return binary\_search(arr, left, mid - 1, key)

else:

return binary\_search(arr, mid + 1, right\_side, key)

else:

return -1

arr = [ 2, 3, 4, 10, 40 ]

find = 10

result = binary\_search(arr, 0, len(arr)-1, find)

if result != -1:

print("The element is present at the index Number: ", str(result))

else:

raise Exception("Invalid Number")

#Task 07

fibo\_store={}

def fibo(n):

if n in fibo\_store:

return fibo\_store[n]

if n==1:

return 0

elif n==2:

return 1

else:

val =fibo(n-1)+fibo(n-2)

fibo\_store[n] = val

return val

for i in range(1,12):

print(fibo(i),end=' ')