Sorting

1] Bubble Sort :

```
def bubbleSort(l):
    n = len(l)

    for i in range(n-1):

        for j in range(n - i-1):
            if l[j] > l[j + 1]:
                l[j], l[j + 1] = l[j + 1], l[j]


l = [10, 8, 20, 5]

bubbleSort(l)

print(*l)
```

OUTPUT :

5 8 10 20

2] Bubble Sort Efficient

```
def bubbleSort(l):

    n=len(l)

    for i in range(n-1):
        swapped=False

        for j in range(n-i-1):
            if l[j]>l[j+1]:
                l[j],l[j+1]=l[j+1],l[j]

                swapped=True

        if swapped==False:
            return

l=[10,8,20,5]

bubbleSort(l)

print(*l)
```

OUTPUT:

5 8 10 20

3] Selection Sort :

```
def selectSort(l):
    n=len(l)

    for i in range(n-1):
        mid_ind=i
        for j in range(i+1,n):
            if l[j]<l[mid_ind]:
                mid_ind=j

        l[mid_ind],l[i]=l[i],l[mid_ind]

l=[10,5,8,20,2,18]

selectSort(l)

print(*l)
```

OUTPUT :

2 5 8 10 18 20

4] Insertion Sort :

```
def insertionSort(l):
    n=len(l)

    for i in range (1,n):
        x=l[i]
        j=i-1

        while j>=0 and x<l[j]:
            l[j+1]=l[j]
            j=j-1

        l[j+1]=x

l=[20,5,40,60,10,30]

insertionSort(l)
print(*l)
```

OUTPUT :

5 10 20 30 40 60

5] Merge two Sorted array :

```
def merge(a,b):
    res=a+b
    res.sort()

    return res

a=[10,15,30]
b=[2,20]

print(*merge(a,b))
```

OUTPUT :

2 10 15 20 30

6] Merge Two Sorted Array Efficient :

```
def merge(a,b):
    res=[]

    m=len(a)
    n=len(b)
    i=j=0

    while i<m and j<n:
        if a[i]<b[j]:
            res.append(a[i])
            i+=1
        else:
            res.append((b[j]))
            j+=1

    while i<m:
        res.append(a[i])
        i+=1

    while j<n:
        res.append(b[j])
        j+=1

    return  res

a=[10,15]
b=[5,6,6,30,40]

print(*merge(a,b))
```

OUTPUT :

5 6 6 10 15 30 40

7] Merge Subarray :

```
def merge(a,low,mid,high):
    left=a[low:mid+1]
    right=a[mid+1:high+1]

    i=j=0
    k=low
    while i<len(left) and j<len(right):
        if left[i]<right[j]:
            a[k]=left[i]
            i+=1
            k+=1

        else:
            a[k]=right[j]
            j+=1
            k+=1

    while i<len(left):
        a[k]=left[i]
        i+=1
        k+=1

    while j<len(right):
        a[k]=right[j]
        j+=1
        k+=1


a=[10,15,20,40,8,11,55]
merge(a,0,3,6)
print(*a)
```

OUTPUT :

8 10 11 15 20 40 55

8] Merge Sort Algorithm :

```
def merge(a,low,mid,high):
    left=a[low:mid+1]
    right=a[mid+1:high+1]

    i=j=0
    k=low

    while i<len(left) and j<len(right):

        if left[i]<right[j]:
            a[k]=left[i]
            k+=1
            i+=1

        else:
            a[k]=right[j]
            k+=1
            j+=1

    while i<len(left):
        a[k]=left[i]
        k+=1
        i+=1

    while j<len(right):
        a[k]=right[j]
        j+=1
        k+=1

def mergeSort(arr,l,r):
    if r>l:
        m=(r+l)//2
        mergeSort(arr,l,m)
        mergeSort(arr,m+1,r)
        merge(arr,l,m,r)

arr=[10,5,30,15,7]

mergeSort(arr,0,4)
print(*arr)
```

OUTPUT :

5 7 10 15 30

9] Partition a Given array :

```
def parition(arr,p):
    n=len(arr)
    arr[p],arr[n-1]=arr[n-1], arr[p]

    temp=[]

    for x in arr:
        if x<=arr[n-1]:
            temp.append(x)

    for x in arr:
        if x>arr[n-1]:
            temp.append(x)

    for i in range(len(arr)):
        arr[i]=temp[i]

arr=[5,13,6,9,12,8,11]

parition(arr,5)

print(arr)
```

OUTPUT :

[5, 6, 8, 13, 9, 12, 11]

10] Lomuto Partition :

```
def lomutoPartition(arr,l,h):
    pivot=arr[h]
    i=l-1

    for j in range(l,h):
        if arr[j]<=pivot:
            i=i+1
            arr[i],arr[j]=arr[j],arr[i]

    arr[i+1],arr[h]=arr[h],arr[i+1]

    return i+1

arr=[10,80,30,90,50,70]

lomutoPartition(arr,0,5)

print(*arr)
```

OUTPUT :

10 30 50 70 80 90

11] Hoare Partition :

```
def hoarePartition(arr,l,h):
    pivot=arr[l]
    i=l-1
    j=h+1

    while True:
        i=i+1
        while arr[i]<pivot:
            i=i+1

        j=j-1
        while arr[j]>pivot:
            j=j-1

        if i>=j:
            return j

        arr[i],arr[j]=arr[j],arr[i]


arr=[5,3,8,4,2,7,1,10]

hoarePartition(arr,0,len(arr)-1)

print(*arr)
```

OUTPUT :

1 3 2 4 8 7 5 10

12] Quick Sort Using Lomuto Partition :

```
def lomutoPartition(arr,l,h):
    pivot=arr[h]
    i=l-1

    for j in range(l,h):
        if arr[j]<=pivot:
            i+=1
            arr[i],arr[j]=arr[j],arr[i]

    arr[i+1],arr[h]=arr[h],arr[i+1]
    return i+1

def qSort(arr,l,h):
    if l<h:
        p=lomutoPartition(arr,l,h)
        qSort(arr,l,p-1)
        qSort(arr,p+1,h)

arr=[8,4,7,9,3,10,5]

qSort(arr,0,6)

print(*arr)
```

OUTPUT :

3 4 5 7 8 9 10

13] Quick Sort Using Hoare Partition :

```
def hoaresPartition(arr,l,h):
    pivot=arr[l]
    i=l-1
    j=h+1

    while True:
        i=i+1
        while arr[i]<pivot:
            i=i+1

        j=j-1
        while arr[j]>pivot:
            j=j-1

        if i>=j:
            return j

        arr[i],arr[j]=arr[j],arr[i]


def qSort(arr,l,h):
    if l<h:
        p=hoaresPartition(arr,l,h)
        qSort(arr,l,p)
        qSort(arr,p+1,h)

arr=[8,4,7,9,3,10,5]

qSort(arr,0,6)

print(*arr)
```

OUTPUT :

3 4 5 7 8 9 10

14] Tail Call Elimination in Quick Sort :

```
def hoarsePartition(arr,l,h):
    pivot=arr[l]

    i=l-1
    j=h+1

    while True:

        i=i+1
        while arr[i]<pivot:
            i=i+1

        j=j-1
        while arr[j]>pivot:
            j=j-1

        if i>=j:
            return j

        arr[i],arr[j]=arr[j],arr[i]


def qSort(arr,l,h):
    while l<h:
        p=hoarsePartition(arr,l,h)
        qSort(arr,l,p)
        l=p+1

arr=[8,4,7,9,3,10,5]

qSort(arr,0,6)

print(*arr)
```

OUTPUT :

3 4 5 7 8 9 10

15] List Sort In Python :

```
#Program 1
print("******************** list **********************")
l1=[5,10,15,1]
l1.sort()
print(l1)

l2=[1,5,3,10]
l2.sort(reverse=True)
print(l2)

l3=['gfg','die','courses']
l3.sort()
print(l3)

def myFun(s):
    return len(s)

l=['gfg','courses','python']
l.sort(key=myFun)
print(l)

l.sort(key=myFun,reverse=True)
print(l)

#Program 2
print("************ sorting user defined using key-fun ***********")
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y

def myFun(p):
    return p.x

l=[Point(1,15),Point(10,5),Point(3,8)]
l.sort(key=myFun)

for i in l:
    print(i.x,i.y)


#Program 3
print("*********** Sorting user Defined using __lt__1 **********")
class Point:
```

```
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def __lt__(self, other):
        return self.x<other.x

l=[Point(1,15),Point(10,5),Point(5,8)]

for i in l:
    print(i.x,i.y)

print("****** sorting user defined using __lt__2 **********")

class Point2:
    def __init__(self,x,y):
        self.x=x
        self.y=y

    def __lt__(self, other):
        if self.x==other.x:
            return self.y<other.y
        else:
            return self.x<other.x

l=[Point2(1,15), Point2(10,5), Point2(1,8)]
l.sort()

for i in l:
    print(i.x,i.y)
```

OUTPUT :

********** list **********

[1, 5, 10, 15]

[10, 5, 3, 1]

['courses', 'die', 'gfg']

['gfg', 'python', 'courses']

['courses', 'python', 'gfg']

********** sorting user defined using key-fun **********

1 15

3 8

10 5

********** Sorting user Defined using __lt__1 **********

1 15

10 5

5 8

****** sorting user defined using __lt__2 **********

1 8

1 15

10 5

16] Sorted in Python :

```
#program 1
print("**************** Program 1 ****************")
l=[10,20,14]
ls=sorted(l)

print(l)
print(ls)

l=[10,-15,-2,1]
ls=sorted(l,key=abs,reverse=True)
print(ls)

#program 2
print("**************** Program 2 ********************")
t=(10,12,5,1)
print(sorted(t))

s={'gfg','courses','python'}
print(sorted(s))

st='gfg'
print(sorted(st))

d={10:'gfg',15:'ide',5:'courses'}
print(sorted(d))
print(d)

l=[(10,15),(1,8),(2,3)]
print(sorted(l))
```

OUTPUT :

***************** Program 1 ******************

[10, 20, 14]

[10, 14, 20]

[-15, 10, -2, 1]

**************** Program 2 *********************

[1, 5, 10, 12]

['courses', 'gfg', 'python']

['f', 'g', 'g']

[5, 10, 15]

{10: 'gfg', 15: 'ide', 5: 'courses'}

[(1, 8), (2, 3), (10, 15)]