
# Data Structures and Algorithm
## Practical Journal

# INDEX

# PRACTICAL NO. 1

**# Q.1: Create Arrays of different data types in python using array module**

```
print("\n24167 - Gautam Ganesh Velu")
print("\nQ.1: Create Arrays of different data types in python using array module")

import array as arr
#integer array
a=arr.array('i',[3,5,8,1,9])
print('\na: ',a)
#traversing using range & array index
for i in range(len(a)):
   print(a[i])

#double array
l2=[1.5,3.7,9.8] #list created
b=arr.array('d',l2)
print('\nb: ',b)
#traversing
for x in b:
   print(x)

#character/Unicode array
c=arr.array('u',"Hello")
print("\nc: ",c)
#traversing
for x in c:
   print(x,end=' ')
```

# OUTPUT:

```
24167 - Gautam Ganesh Velu

Q.1: Create Arrays of different data types in python using array module

a:  array('i', [3, 5, 8, 1, 9])
3
5
8
1
9

b:  array('d', [1.5, 3.7, 9.8])
1.5
3.7
9.8
D:\Programmings in College\DSA programming\Arrays\Pract 1.py:29: DeprecationWarning: The 'u' type code is deprecated and will be removed in
Python 3.16
  c=arr.array('u',"Hello")

c:  array('u', 'Hello')
H e l l o
```

**Q 2: Implement different possible operations on Array in python using array module**

```
print("\nQ.2: Implement different possible operations on Array in python using array module")
import array as arr

#integer array
a=arr.array('i',[3,5,8,1,9])
print('a: ',a)

#traversing : Visiting all the elements of array one by one
for i in range(len(a)):
   print(a[i])

#Accessing Individual element
print("\nElement at index 2 is: ",a[2])

#Inserting element in to the array [using insert method]
a.insert(3,60) #inserting element 60 at position/index 3
print("array after insertion of 60 at pos 3 : ",a)

#append() method can be used to insert element at end
a.append(90)
print("array after appending 90: ",a)

#Removing any element/Deletion of particular element from array
a.remove(8) #remove method can be used to remove particular element from array
print('Array after removal of 8 : ',a)

del a[2] #del can be used to delete element at particular index
print("Array after deletion of element at index 2: ",a)
```

```
#Pop() method can be used to delete the last element
#Pop(index) method can be used to delete the element at given index
a.pop()
print("Array after calling pop(): ",a)

#Search
#index(ele) method can be used to search the given element
# if it is present then it will return the index or it will generate error
print("\nElement 1 present at index: ",a.index(1))
```

## OUTPUT:

```
Q.2: Implement different possible operations on Array in python using array module
a:  array('i', [3, 5, 8, 1, 9])
3
5
8
1
9

Element at index 2 is:  8
array after insertion of 60 at pos 3 :  array('i', [3, 5, 8, 60, 1, 9])
array after appending 90:  array('i', [3, 5, 8, 60, 1, 9, 90])
Array after removal of 8 :  array('i', [3, 5, 60, 1, 9, 90])
Array after deletion of element at index 2:  array('i', [3, 5, 1, 9, 90])
Array after calling pop():  array('i', [3, 5, 1, 9])

Element 1 present at index:  2
```

# PRACTICAL NO. 2

**Q 1: Create 1_D Array & Implement different possible operations on Array in python using numpy module**

```
print("\n24167 - Gautam Ganesh Velu")

print("\nQ.1: Create 1_D Array & Implement different possible operations on Array in python using numpy module")


import numpy as np


#Creating integer array

a=np.array([3,5,8,1,9])

print('a: ',a)

#traversing : Visiting all the elements of array one by one

for i in range(len(a)):

   print(a[i])


#Accessing Individual element

print("\nElement at index 2 is: ",a[2])

#Negative indexing

print("\nLast element of array is: ",a[-1])


#Inserting element in to the array [using insert method]

a=np.insert(a,3,60) #inserting element 60 at position/index 3

print("array after insertion of 60 at pos 3 : ",a)

#append() method can be used to insert element at end

a=np.append(a,90)

print("array after appending 90: ",a)


#Removing any element/Deletion of particular element from array

a=np.delete(a,4) #delete method can be used to remove element at particular index from array
```

```
print('Array after removal of 4th index element : ',a)


#Search

#where(ele) method can be used to search the given ele if it is present then it will return the index or it will generate error

print("\nElement 3 present at index: ",np.where(a==3))


#Sorting

print("\nSorted array is: ",np.sort(a))
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU

Q.1: Create 1_D Array & Implement different possible operations on Array in python using numpy module
a:  [3 5 8 1 9]
3
5
8
1
9

Element at index 2 is:  8

Last element of array is:  9
array after insertion of 60 at pos 3 :  [ 3  5  8 60  1  9]
array after appending 90:  [ 3  5  8 60  1  9 90]
Array after removal of 4th index element :  [ 3  5  8 60  9 90]

Element 3 present at index:  (array([0]),)

Sorted array is:  [ 3  5  8  9 60 90]
```

**Q 2: Create 2_D Array[Matrix] & perform some operations[Access any particular element, Display matrix ,update element] using list it python**

```
print("\nQ.2: Create 2_D Array[Matrix] & perform some operations[Access any particular element, Display matrix ,update element] using list it python ")

#Creation of matrix

mat1=[[1,2,3],[2,5,7],[7,9,8]]  #Here list if lists represent a 2-D array i.e. matrix


#Display matrix

print("\nThe matrix is:\n ")

for i in range(len(mat1)):

   for j in range(len(mat1[0])):

      print(mat1[i][j],end=' ')

   print()


#Accesing element at particular index

print('Element present at index[0,1] is: ',mat1[0][1])


#Updating matrix element

mat1[0][0]=6

print("\nAfter updating 0,0 th element to 6, matrix will be\n")


for i in range(len(mat1)):

   for j in range(len(mat1[0])):

      print(mat1[i][j],end=' ')

   print()
```

## OUTPUT:

```
Q.2: Create 2_D Array[Matrix] & perform some operations[Access any particular element, Display matrix ,update element] using list it python

The matrix is:

1 2 3
2 5 7
7 9 8
Element present at index[0,1] is:  2

After updating 0,0 th element to 6, matrix will be

6 2 3
2 5 7
7 9 8
```

**Q 3: Matrix addition program using list in python [Operations on 2-D array]**

```
print("\nQ.3: Matrix addition program using list in python  [Operations on 2-D array]")

#Display matrix
def display_matrix(a):
    for i in range(len(a)):
      for j in range(len(a[0])):
         print(a[i][j],end=' ')
      print()

def addition(a,b):
   c=[]
   for i in range(len(a)):
      tmp=[]
      for j in range(len(a[0])):
         tmp.append(a[i][j]+b[i][j])

      c.append(tmp)
   return c

#main
mat1=[[1,2,3],[2,5,7],[7,9,8]]  #Here list of lists represent a 2-D array i.e. matrix
mat2=[[1,5,3],[2,1,1],[8,1,3]]

print("\nThe first matrix is: ")
display_matrix(mat1)
print("\nThe Second matrix is: ")
display_matrix(mat2)

mat3=addition(mat1,mat2)
```

```
print("\nMatrix addition is: ")
for i in range(len(mat3)):
    for j in range(len(mat3[0])):
        print(mat3[i][j],end=' ')
    print()
```

## OUTPUT:

```
Q.3: Matrix addition program using list in python  [Operations on 2-D array]

The first matrix is:
1 2 3
2 5 7
7 9 8

The Second matrix is:
1 5 3
2 1 1
8 1 3

Matrix addition is:
2 7 6
4 6 8
15 10 11
```

# PRACTICAL NO. 3

**Q 1: Create Matrix using numpy module & do matrix operations (addition, Subtraction, Multiplication & Transpose) in python.**

```
print("24167 - GAUTAM GANESH VELU")

print("Q 1: Create Matrix using numpy module & do matrix operations  (addition, Subtraction, Multiplication & Transpose) in python.")


import numpy as np

a=np.array([[1,2,3],[4,5,6],[7,8,9]])

b=np.array([[1,2,1],[4,3,2],[6,3,2]])

print("\nMatrix1: ")

print(a)

print("\nMatrix2: ")

print(b)


#matrix addition

c=a+b

print("\nMatrix addition is: ")

print(c)


#matrix Subtraction

d=a-b

print("\nMatrix Subtraction is: ")

print(d)


#matrix Multiplication

e=a@b

print("\nMatrix Multiplication is: ")

print(e)
```

```
#matrix transpose

t=a.transpose()

print("\nTranspose of Matrix a is: ")

print(t)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q 1: Create Matrix using numpy module & do matrix operations  (addition, Subtraction, Multiplication & Transpose) in python.

Matrix1:
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Matrix2:
[[1 2 1]
 [4 3 2]
 [6 3 2]]

Matrix addition is:
[[ 2  4  4]
 [ 8  8  8]
 [13 11 11]]

Matrix Subtraction is:
[[0 0 2]
 [0 2 4]
 [1 5 7]]

Matrix Multiplication is:
[[27 17 11]
 [60 41 26]
 [93 65 41]]

Transpose of Matrix a is:
[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

**Q 2: Create Sparse Matrix using scipy.sparse module & apply different methods**

```
print(“24167 – GAUTAM GANESH VELU”)
print("Q 2: Create Sparse Matrix using scipy.sparse module & apply different methods")

import numpy as np
from scipy.sparse import csr_array

#Creating Normal 2-D array using numpy module
A=np.array([[1,2,0],[0,0,0],[0,0,2]])

#Converting to sparse matrix
SA=csr_array(A)
print("Original Matrix is:\n ");
print(A)
print("\nThe equivalent sparse matrix is:\n")
print(SA)

#Apply properties & method on sparse matrix
#Viewing stored data (not the zero items) with the data property:
print("Data in the sparse matrix is : ",SA.data)

#Counting nonzeros with the count_nonzero() method:
print("Number of Non-zero elements is :",SA.count_nonzero())

#Converting from csr to csc with the tocsc() method:
new_mat=SA.tocsc()
print("\nEquivalent CSC matrix is: ",new_mat)
```

**OUTPUT:**

```
24167 - GAUTAM GANESH VELU
Q 2: Create Sparse Matrix using scipy.sparse module & apply different methods
Original Matrix is:

[[1 2 0]
 [0 0 0]
 [0 0 2]]

The equivalent sparse matrix is:

<Compressed Sparse Row sparse array of dtype 'int64'
        with 3 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (0, 1)        2
  (2, 2)        2
Data in the sparse matrix is :  [1 2 2]
Number of Non-zero elements is : 3

Equivalent CSC matrix is:  <Compressed Sparse Column sparse array of dtype 'int64'
        with 3 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (0, 1)        2
  (2, 2)        2
```

# PRACTICAL NO. 4

**Q 1: Create sparse matrix & display its transpose in python**

```
print("24167 - GAUTAM GANESH VELU")
print("Q 1: Create sparse matrix & display its transpose in python")

#sparse matrix implementation in python [Create & Transpose ]
import numpy as np
from scipy.sparse import csr_matrix
a=np.array([[1,0,0],[0,0,4],[0,2,0]])
sm=csr_matrix(a)
print("\nThe simple matrix is: ")
print(a)
print("\n The equivalent csr sparce matrix is")
print(sm)

#sparse matrix transpose
b=sm.transpose().tocsr()
print("Transposed matrix is: ")
print(b)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q 1: Create sparse matrix & display its transpose in python

The simple matrix is:
[[1 0 0]
 [0 0 4]
 [0 2 0]]

 The equivalent csr sparce matrix is
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 3 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (1, 2)        4
  (2, 1)        2
Transposed matrix is:
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 3 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (1, 2)        2
  (2, 1)        4
```

**Q2. Create a python program for addition of 2 sparse matrices**

```
print("24167 - GAUTAM GANESH VELU")
print("\nQ 2: Create a python program for addition of 2 sparse matrices")

#sparse matrix implementation in python [Create & Transpose ]
import numpy as np
from scipy.sparse import csr_matrix
a=np.array([[1,0,0],[0,0,4],[0,2,0]])
b=np.array([[0,0,1],[0,0,0],[0,3,0]])
sm1=csr_matrix(a)
sm2=csr_matrix(b)
print("\nThe First simple matrix is: ")
print(a)
print("\n The equivalent csr sparce matrix is")
print(sm1)
print("\nThe Second simple matrix is: ")
print(b)
print("\n The equivalent csr sparce matrix is")
print(sm2)

#addition
c=sm1+sm2
print("\nAddition of sparse matrices is: ")
print(c)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU

Q 2: Create a python program for addition of 2 sparse matrices

The First simple matrix is:
[[1 0 0]
 [0 0 4]
 [0 2 0]]

 The equivalent csr sparce matrix is
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 3 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (1, 2)        4
  (2, 1)        2

The Second simple matrix is:
[[0 0 1]
 [0 0 0]
 [0 3 0]]

 The equivalent csr sparce matrix is
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 2 stored elements and shape (3, 3)>
  Coords        Values
  (0, 2)        1
  (2, 1)        3

Addition of sparse matrices is:
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 4 stored elements and shape (3, 3)>
  Coords        Values
  (0, 0)        1
  (0, 2)        1
  (1, 2)        4
  (2, 1)        5
```

# PRACTICAL NO. 5

**Q1. Implement Singly Linked List data structure in python**

```
print("24167 - GAUTAM GANESH VELU")

print("Q1. Implement Singly Linked List data structure in python")

class Node:
  def __init__(self,data):
    self.data=data
    self.next=None

class List:
  def __init__(self):
    self.head=None

  def insert_at_beg(self,data):
    new_node=Node(data)

    if(self.head==None):
      self.head=new_node
    else:
      new_node.next=self.head
      self.head=new_node

  def insert_at_pos(self,data,p):
    if(p==0):
      self.insert_at_beg(data)
    else:
      i=0
      tmp=self.head
```

```
        while(i<p-1 and tmp is not None):
            tmp=tmp.next
            i=+1

        if(tmp==None):
            print("Invalid Position")
        else:
            new_node=Node(data)
            new_node.next=tmp.next
            tmp.next=new_node

    def insert_at_end(self,data):
        new_node=Node(data)
        if(self.head==None):
            self.head=new_node
        else:
            tmp=self.head
            while(tmp.next!=None):
                tmp=tmp.next

            tmp.next=new_node

    def display(self):
        if(self.head==None):
            print('empty List !!')
        else:
            tmp=self.head
            while(tmp):
                print(tmp.data, end="-->")
                tmp=tmp.next
            print("None")
```

```
def del_first(self):
  if(self.head==None):
    print("Empty List")
  else:
    tmp=self.head
    self.head=tmp.next
    print("Deleted", tmp.data)
def del_last(self):
  if(self.head==None):
      print("Empty List!!")
  else:
    tmp=self.head
    while(tmp.next):
      prev=tmp
      tmp=tmp.next
    prev.next=None
    print("Deleted", tmp.data)
def del_at_pos(self,p):
  if(self.head==None):
    print("Empty List!!")
  else:
    if(p==0):
      self.del_first()
    else:
      i=0
      tmp=self.head
      while(i<p and tmp is not None):
        pre=tmp
        tmp=tmp.next
        i=+1
      if(tmp==None):
```

```
            print("invalid Position")
        else:
            pre.next=tmp.next
            tmp.next=None
            print("Deleted",tmp.data)
if(__name__=="__main__"):
  l=List()
  while(True):
    print("\n1.Insert at Beginning\n2.Insert at End\n3.Insert at position\n4.Delete First\n5.Delete Last\n6.Delete at position\n7.Display\n8.Exit")
    ch=int(input("\nEnter any choice:"))
    if(ch==8):
      break
    if(ch==1):
      d=input("\nEnter the data: ")
      l.insert_at_beg(d)

    elif(ch==2):
      d=input("\nEnter the data")
      l.insert_at_end(d)

    elif(ch==3):
      d=int(input("\nEnter the data"))
      p=int(input("\nEnter the position: "))
      l.insert_at_pos(d,p)

    elif(ch==4):
      l.del_first()
    elif(ch==5):
      l.del_last()
    elif(ch==6):
```

```
        p=int(input("\nEnter the position: "))
        l.del_at_pos(p)

    elif(ch==7):
        l.display()

    else:
        print("Invalid Option!!")
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Implement SLL data structure in python

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:1

Enter the data: 5

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:2

Enter the data8

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit
```

```
Enter any choice:3

Enter the data2

Enter the position: 1

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:7
5-->2-->8-->None

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:4
Deleted 5

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:7
2-->8-->None

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:5
Deleted 8

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice:7
2-->None

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit
```

# PRACTICAL NO. 6

**Q1. Implement Doubly Linked List data structure in python**

```
print("24167 - GAUTAM GAENSH VELU")
print("Q1. Implement Doubly Linked List data structure in python")

#Doubly Linked List Implementation
class Node:
  def __init__(self,data):
    self.data=data
    self.next=None
    self.prev=None


class DList:

  def __init__(self):
    self.head=None

  def insert_at_beg(self,data):
    new_node=Node(data)

    if(self.head==None):
      self.head=new_node
    else:
      new_node.next=self.head
      self.head.prev=new_node
      self.head=new_node

  def insert_at_pos(self,data,p):
```

```
    if(p==0):
        self.insert_at_beg(data)
    else:
        i=0
        tmp=self.head
        while(i<p-1 and tmp is not None):
            tmp=tmp.next
            i=i+1
        if(tmp == None):
            print("Invalid Position")
        else:
            new_node=Node(data)
            new_node.next=tmp.next
            tmp.next=new_node
            new_node.prev=tmp
            tmp.next.prev=new_node

def insert_at_end(self,data):
    new_node=Node(data)
    if(self.head==None):
        self.head=new_node
    else:
        tmp=self.head
        while(tmp.next!=None):
            tmp=tmp.next
        tmp.next=new_node
        new_node.prev=tmp

def display(self):
    if(self.head == None):
        print("Empty List !!")
```

```
    else:
        tmp=self.head
        while(tmp):
            print(tmp.data ,end="-->")
            tmp=tmp.next
        print("None")

def del_first(self):
    if(self.head==None):
        print("Empty List")
    elif (self.head.next is None):
        print("Deleted ",self.head.data)
        self.head=None
    else:
        tmp=self.head
        self.head=tmp.next
        self.head.prev=None
        print("Deleted ",tmp.data)

    def del_last(self):
        if(self.head==None):
            print("Empty List !!")
        else:
            tmp=self.head
            while(tmp.next):
                pnode=tmp
                tmp=tmp.next
            pnode.next=None
            print("Deleted ",tmp.data)

def del_at_pos(self,p):
```

```
    if(self.head == None):
      print("Empty List !!")
    else:
      tmp=self.head
      if(p == 0):
        self.del_first()
      else:
        i=0
        while(i<p and tmp is not None):
          tmp=tmp.next
          i+=1
        if(tmp==None):
          print("Invalid Position")
        elif(tmp.next is None):
          self.del_last()
        else:
          pnode=tmp.prev
          nnode=tmp.next
          pnode.next=nnode
          nnode.prev=pnode
          print("Deleted ",tmp.data)

if(__name__== "__main__"):
  l=DList()
  while(True):
    print("\n1.Insert at Beginning\n2.Insert at End\n3.Insert at position\n4.Delete
First\n5.Delete Last\n6.Delete at position\n7.Display\n8.Exit")

    ch=int(input("\nEnter any choice: "))
    if(ch==8):
      break
```

```
if(ch==1):
   d=input("\nEnter the data: ")
   l.insert_at_beg(d)
elif(ch==2):
   d=input("\nEnter the data: ")
   l.insert_at_end(d)
elif(ch==3):
   d=int(input("\nEnter the data"))
   p=int(input("\nEnter the position: "))
   l.insert_at_pos(d,p)
elif(ch==4):
   l.del_first()
elif(ch==5):
   l.del_last()
elif(ch==6):
   p=int(input("\nEnter the position: "))
   l.del_at_pos(p)
elif(ch==7):
   l.display()
else:
   print("Invalid option!!")
```

## OUTPUT:

```
24167 - GAUTAM GAENSH VELU
Q1. Implement Doubly Linked List data structure in python

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 7

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 9

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 5

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 0

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 3

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
```

```
6.Delete at position
7.Display
8.Exit

Enter any choice: 7
3-->0-->5-->9-->7-->None

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 8
PS D:\Programmings in College\DSA programming\Linked List>
```

# PRACTICAL NO. 7

**Q1. Implement Singly Circular Linked List data structure in python**

```
print(“24167 – GAUTAM GANESH VELU”)
print("Q1. Implement Singly Circular Linked List data structure in python")
#SCLL Implementation
class Node:
  def __init__(self,data):
    self.data=data
    self.next=None

class CSList:
  def __init__(self):
    self.head=None
    self.tail=None

  def insert_at_beg(self,data):
    new_node=Node(data)
    if(self.head == None):
      self.head=new_node
      self.tail=new_node
      self.tail.next=self.head
    else:
      new_node.next=self.head
      self.tail.next=new_node
      self.head=new_node

  def insert_at_pos(self,data,p):
    if(p==0):
      self.insert_at_beg(data)
      return
```

```
    new_node=Node(data)
    i=0
    tmp=self.head
    for i in range(p-1):
        tmp=tmp.next
        if(tmp is None):
            break
    if(tmp is not None):
        new_node.next=tmp.next
        tmp.next=new_node

def insert_at_end(self,data):
    new_node=Node(data)
    if(self.head==None):
        self.head=new_node
        self.tail=new_node
        new_node.next=self.head
    else:
        self.tail.next=new_node
        self.tail=new_node
        self.tail.next=self.head

def display(self):
    if(self.head == None):
        print("Empty List !!")
    else:
        tmp=self.head
        while(tmp.next!=self.head):
            print(tmp.data ,end="-->")
            tmp=tmp.next
        print(tmp.data ,end="-->")
```

```
        print("None")

    def del_first(self): #check
        if(self.head==None):
            print("Empty List")
        elif(self.head==self.tail):
            print("Deleted ",self.head.data)
            self.head=None
            self.tail=None
        else:
            tmp=self.head
            print("Deleted ",tmp.data)
            self.head=self.head.next
            self.tail.next=self.head

    def del_last(self):
        if(self.head==None):
            print("Empty List !!")
        elif(self.head.next==self.head):
            print("Deleted ",self.head.data)
            self.head=None
            self.tail=None
        else:
            tmp=self.head
            while(tmp.next!=self.tail):
                tmp=tmp.next
        print("Deleted ",self.tail.data)
        self.tail=tmp
        self.tail.next=self.head

    def del_at_pos(self,p):
```

```
    if(self.head == None):
      print("Empty List !!")
    else:
      if(p == 0):
        self.del_first()
      else:
        tmp=self.head
        pnode=tmp
      for i in range(p):
        if(tmp is None):
          break
        pnode=tmp
        tmp=tmp.next
      if(tmp is None):
        print("Invalid Position")
      else:
        pnode.next=tmp.next
        print("Deleted ",tmp.data)

if(__name__== "__main__"):
  l=CSList()
while(True):
  print("\n1.Insert at Beginning\n2.Insert at End\n3.Insert at position\n4.Delete
First\n5.Delete Last\n6.Delete at position\n7.Display\n8.Exit")
  ch=int(input("\nEnter any choice: "))
  if(ch==8):
    break;
  if(ch==1):
    d=input("\nEnter the data: ")
    l.insert_at_beg(d)
  elif(ch==2):
```

```
    d=input("\nEnter the data: ")
    l.insert_at_end(d)
elif(ch==3):
    d=int(input("\nEnter the data"))
    p=int(input("\nEnter the position: "))
    l.insert_at_pos(d,p)
elif(ch==4):
    l.del_first()
elif(ch==5):
    l.del_last()
elif(ch==6):
    p=int(input("\nEnter the position: "))
    l.del_at_pos(p)
elif(ch==7):
    l.display()
else:
    print("Invalid option!!")
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Implement Singly Circular Linked List data structure in python

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 666

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 9

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 1

Enter the data: 8

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 7
8-->9-->666-->None

1.Insert at Beginning
2.Insert at End
3.Insert at position
4.Delete First
5.Delete Last
6.Delete at position
7.Display
8.Exit

Enter any choice: 8
```

# PRACTICAL NO. 8

**Q1. Write a program to reverse a linked list in python**

```
print("24167 - GAUTAM GANESH VELU")

print("Q1. Write a program to reverse a linked list in python")

class Node:
  def __init__(self,data):
    self.data=data
    self.next=None

class List:
  def __init__(self):
    self.head=None

  def create(self):
    n=int(input("How many elements you want to insert?\n"))
    for i in range(n):
      d=int(input("\nEnter data:"))
      new_node=Node(d)
      if(self.head==None):
        self.head=new_node
      else:
        tmp=self.head
        while(tmp.next):
          tmp=tmp.next
        tmp.next=new_node

  def display(self):
```

```
        if(self.head == None):
            print("Empty List!")
        else:
            tmp=self.head
            while(tmp):
                print(tmp.data,end="-->")
                tmp=tmp.next

    def reverse(self):
        prev=None
        current=self.head
        while(current is not None):
            next=current.next
            current.next=prev
            prev=current
            current=next
        self.head=prev

L=List()
L.create()
print("\nThe List is: ")
L.display()
print("\nThe reversed list is: ")
L.reverse()
L.display()
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Write a program to reverse a linked list in python
How many elements you want to insert?
6

Enter data:4

Enter data:6

Enter data:9

Enter data:2

Enter data:0

Enter data:1

The List is:
4-->6-->9-->2-->0-->1-->
The reversed list is:
1-->0-->2-->9-->6-->4-->
```

# PRACTICAL NO. 9

**Q1. Implement Stack Static data structure in python**

```
print("24167 - GAUTAM GANESH VELU")
print("Q1. Implement Stack Static data structure in python")


class Stack:
  def __init__(self,capacity):
    self.capacity= capacity
    self.stack = [None]*capacity
    self.top = -1

  def push(self, item):
    if self.is_full():
      print("Stack Overflow!")
      return
    self.top +=1
    self.stack[self.top]=item

  def pop(self):
    if self.is_empty():
      print("Stack Underflow!")
      return None
    item = self.stack[self.top]
    self.stack[self.top]=None
    self.top-=1
    return item

  def peek(self):
    if self.is_empty():
      return None
    return self.stack[self.top]
```

```
    def is_empty(self):
        return self.top ==-1

    def is_full(self):
        return self.top == self.capacity - 1

    def display(self):
        if self.is_empty():
            print("Stack Underflow")
            return
        for i in range(self.top, -1, -1):
            print(self.stack[i])
        print("\n")

s=Stack(5)
while(1):
    print("\n1.Push\n2.Pop\n3.Peek\n4.Display\n5.Exit")
    ch=int(input("\nEnter yoour choice: "))
    if(ch==5):
        break
    if(ch==1):
        n=input("enter the item to be pushed: ")
        s.push(n)
    elif(ch==2):
        print("Poped element is: ",s.pop())
    elif(ch==3):
        print("Peeked element is: ",s.peek())
    else:
        s.display()
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Implement Stack Static data structure in python

1.Push
2.Pop
3.Peek
4.Display
5.Exit

Enter yoour choice: 1
enter the item to be pushed: 4

1.Push
2.Pop
3.Peek
4.Display
5.Exit

Enter yoour choice: 1
enter the item to be pushed: 8

1.Push
2.Pop
3.Peek
4.Display
5.Exit

Enter yoour choice: 1
enter the item to be pushed: 2
```

```
1.Push
2.Pop
3.Peek
4.Display
5.Exit

Enter yoour choice: 4
2
8
4
```

# PRACTICAL NO. 10

**Q1. Implement Stack Dynamic data structure in python**

```
print("24167 - GAUTAM GANESH VELU")
print("Q1. Implement Stack Dynamic data structure in python")


class Node:
  def __init__(self, data):
    self.data = data
    self.next = None
class Stack:
  def __init__(self):
    self.top = None
  def is_empty(self):
    return  self.top is None

  def push(self, data):
    new_node  =Node(data)
    if self.top is None:
      self.top = new_node
    else:
      new_node.next = self.top
      self.top = new_node

  def pop(self):
    if  self.is_empty():
      print("\nStack Underflow")
    else:
      temp = self.top.data
      self.top = self.top.next
```

```
            print("Deleted  element is: ",temp)

    def peek(self):
        if self.is_empty():
            print("\nStack is empty")
        else:
            print("The topmost element of stack is: ",self.top.data)

    def display(self):
        if self.is_empty():
            print("\nStack is Empty!")
        else:
            tmp=self.top
            while(tmp!= None):
                print(tmp.data)
                tmp = tmp.next
st = Stack()
while True:
    print("\n1.Push\n2.Pop\n3.Peek\n4.Display\n5.Exit")
    ch=int(input("Enter your choice: "))
    if ch == 5:
        break
    if ch == 1:
        n=input("\nEnter the element to be pushed: ")
        st.push(n)
    elif ch == 2:
        st.pop()
    elif ch == 3:
        st.peek()
    else:
        st.display()
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Implement Stack Dynamic data structure in python

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element to be pushed: 4

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element to be pushed: 8

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element to be pushed: 4

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 1

Enter the element to be pushed: 6

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 2
Deleted  element is:  6

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 3
The topmost element of stack is:  4

1.Push
2.Pop
3.Peek
4.Display
5.Exit
Enter your choice: 4
4
8
4
```

# PRACTICAL NO. 11

**Q1. Static implementation of queue data structure in python using list**

```
print("24167 - GAUTAM GANESH VELU")
print("Q1. Static implementation of queue data structure in python using list")


class Queue:
  def __init__(self, capacity):
    self.front = -1
    self.rear = -1
    self.capacity = capacity
    self.que = [None] * capacity

  # Function to insert an element at the rear of the queue
  def enqueue(self, data):
  # Check if the queue is full
    if self.rear == self.capacity - 1:
      print("Queue is full")
      return
    if self.front==-1:
      self.front+=1

    # Insert element at the rear
    self.rear += 1
    self.que[self.rear] = data

  # Function to delete an element from the front of the queue
  def dequeue(self):
  # If the queue is empty
    if self.front==-1:
```

```
            print("Queue is empty")
            return
        elif (self.front==self.rear):
            item =self.que[self.front]
            self.front=self.rear=-1
            print("Deleted element is: ",item)
        else :
            item =self.que[self.front]
            self.front+=1
            print("Deleted element is: ",item)

    # Function to print queue elements
    def display(self):
        if self.front==-1:
            print("Queue is Empty")
            return

        # Traverse front to rear and print elements
        for i in range(self.front, self.rear + 1):
            print(self.que[i], end=" <-- ")
        print()

    # Function to print the front of the queue
    def front_element(self):
        if self.front == -1:
            print("Queue is Empty")
            return
        print("Front Element is:", self.que[self.front])

# Driver code
if __name__ == "__main__":
```

```
# Create a queue of capacity 4
q = Queue(4)
# Print queue elements
q.display()
# Insert elements in the queue
q.enqueue(20)
q.enqueue(30)
q.enqueue(40)
q.enqueue(50)
# Print queue elements
q.display()
# Insert element in the queue
q.enqueue(60)
# Print queue elements
q.display()
# Dequeue elements
q.dequeue()
q.dequeue()
print("After two node deletions")
# Print queue elements
q.display()
print("After one insertion")
q.enqueue(60)
# Print queue elements
q.display()
# Print front of the queue
q.front_element()
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Static implementation of queue data structure in python using list
Queue is Empty
20 <-- 30 <-- 40 <-- 50 <--
Queue is full
20 <-- 30 <-- 40 <-- 50 <--
Deleted element is:  20
Deleted element is:  30
After two node deletions
40 <-- 50 <--
After one insertion
Queue is full
40 <-- 50 <--
Front Element is: 40
```

# PRACTICAL NO. 12

**Q1. Python program to implement a Queue using singly linked list [Dynamic Implementation]**

```
print("24167 - GAUTAM GANESH VELU")

print("Q1. Python program to implement a Queue using singly linked list [Dynamic Implementation]")

# Class representing a node in the class

class Node:

  def __init__(self,data):

    self.data = data

    self.next = None


# Class to implement stack using a singly linked list

class Queue:

  def __init__(self):

    self.front = None

    self.rear=None


  # Function to check if the stack is empty

  def is_empty(self):

    # If head is None, the stack is empty

    return self.front is None


  # Function to push an element onto the stack

  def enqueue(self,data):

    # Create a new node with given data

    new_node = Node(data)

    if self.front is None:

      self.front =self.rear= new_node

    else:

      self.rear.next=new_node
```

```
            self.rear=new_node

    # Function to remove the top element from the stack
    def dequeue(self):
        # Check for stack underflow
        if self.is_empty():
            print("\nQueue is empty")
        else:
            temp = self.front.data
            self.front = self.front.next
            print("Deleted element is: ",temp)

    # Function to display the contents of the stack
    def display(self):
        if self.is_empty():
            print("\nQueue is empty!")
        else:
            tmp=self.front
            while(tmp !=None):
                print(tmp.data,end=" ")
                tmp=tmp.next
    # Creating a stack
Q = Queue()
while True:
    print("\n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit")
    ch=int(input("Enter your choice: "))
    if ch==4:
        break
    if ch==1:
        n=input("\nEnter the element to be enqueued: ")
        Q.enqueue(n)
```

```
elif ch==2:
    Q.dequeue()
else:
    Q.display()
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Python program to implement a Queue using singly linked list [Dynamic Implementation]

1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element to be enqueued: 4

1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element to be enqueued: 8

1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element to be enqueued: 7
```

```
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 1

Enter the element to be enqueued: 2

1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 2
Deleted element is:  4

1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 3
8 7 2
1.Enqueue
2.Dequeue
3.Display
4.Exit
Enter your choice: 4
```

# PRACTICAL NO. 13

**Q1. Python program to reverse a string using stack**

```
print("24167 - GAUTAM GANESH VELU")
print("Q1. Python program to to reverse a string using stack")


# Class representing a node in the class
class Node:
  def __init__(self,data):
    self.data = data
    self.next = None

# Class to implement stack using a singly linked list
class Stack:
  def __init__(self):
    self.top = None

  # Function to check if the stack is empty
  def is_empty(self):
    # If head is None, the stack is empty
    return self.top is None

  # Function to push an element onto the stack
  def push(self,data):
    # Create a new node with given data
    new_node = Node(data)
    if self.top is None:
      self.top = new_node
    else:
      new_node.next = self.top
      self.top=new_node
```

```
    # Function to remove the top element from the stack
    def pop(self):
        if not self.is_empty():
            temp = self.top.data
            self.top = self.top.next
            return temp

# Creating a stack
st = Stack()
str=input("\nEnter the string to be reversed: ")
print("\nOriginal String is: ",str)

#String reversal
#Pushing individual characters from string into the stack
for ch in str:
    st.push(ch)
rlist=[]
while (not st.is_empty()):
    rlist.append(st.pop())
rstr=''.join(rlist)
print("\nReversed String is: ",rstr)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Python program to to reverse a string using stack

Enter the string to be reversed: Data Structure And Algorithm

Reversed String is:  mhtiroglA dnA erutcurtS ataD
```

# PRACTICAL NO. 14

**Q1. Python program to evaluate postfix expression using stack**

```
print("24167 - GAUTAM GANESH VELU")
print("Q1. Python program to evaluate postfix expression using stack")


# Class representing a node in the class
class Node:
  def __init__(self,data):
    self.data = data
    self.next = None
# Class to implement stack using a singly linked list
class Stack:
  def __init__(self):
    self.top = None
  # Function to check if the stack is empty

  def is_empty(self):
    # If head is None, the stack is empty
    return self.top is None

  # Function to push an element onto the stack
  def push(self,data):
    # Create a new node with given data
    new_node = Node(data)
    if self.top is None:
      self.top = new_node
    else:
      new_node.next = self.top
      self.top=new_node
```

```
    # Function to remove the top element from the stack
    def pop(self):
        if not self.is_empty():
            temp = self.top.data
            self.top = self.top.next
            return temp

    # Function to evaluate postfix expression
def evaluate_postfix(expr):
    st=Stack()
    for ch in expr:
        if ch.isdigit():
            st.push(int(ch))
        else:
            r_operand=st.pop()
            l_operand=st.pop()
            if(ch=='+'):
                result=l_operand+r_operand
            elif(ch=='-'):
                result=l_operand-r_operand
            elif(ch=='*'):
                result=l_operand*r_operand
            elif(ch=='/'):
                result=l_operand/r_operand
            else:
                print("\nUnsupported operator")
            st.push(result)
    return(st.pop())

postfix_expr="333+*"
```

```
result=evaluate_postfix(postfix_expr)

print("Result: ",result)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Python program to evaluate postfix expression using stack
Result:  18
```

# PRACTICAL NO. 15

**Q1. BST(Binary Search tree) Implementation (Insert, Traversals(Inorder, Preorder, Postorder), search)**

```
print("24167 - GAUTAM GANESH VELU")

print("Q1. Q1. BST(Binary Search tree) Implementation (Insert, Traversals(Inorder, Preorder, Postorder), search)")


class BST:
  def __init__(self,data):
    self.data=data
    self.left=None
    self.right=None
  def insert(self,data):
    if(self.data == data ):
      return
    elif(data<self.data):
      if(self.left==None):
        self.left=BST(data)
      else:
        self.left.insert(data)
    else:
      if(self.right==None):
        self.right=BST(data)
      else:
        self.right.insert(data)


  def inorder(self):
    l=[]
    if(self.left):
      l+=self.left.inorder()
```

```
    l.append(self.data)
    if(self.right):
        l+=self.right.inorder()
    return l

def preorder(self):
    l=[]
    l.append(self.data)
    if(self.left):
        l+=self.left.inorder()
    if(self.right):
        l+=self.right.inorder()
    return l

def postorder(self):
    l=[]
    if(self.left):
        l+=self.left.inorder()
    if(self.right):
        l+=self.right.inorder()
    l.append(self.data)
    return l

def search(self,value):
    if(value==self.data):
        return True
    if(value < self.data):
        if(self.left):
            return self.left.search(value)
        if(value>self.data):
            if(self.right):
```

```
            return self.right.search(value)
        return False

def create(lst):
  root=BST(lst[0])
  for i in range(1,len(lst)):
    root.insert(lst[i])
  return root

my_list=[23,56,78,21,40]
b=create(my_list)

in_list=b.inorder()
print("Inorder: ",in_list)

pre_list=b.preorder()
print("Preorder: ",pre_list)

post_list=b.postorder()
print("Postorder: ",post_list)

s=int(input("Enter the value to be searched: "))

if(b.search(s)):
  print("Element present")
else:
    print("Element Not present")
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Q1. Q1. BST(Binary Search tree) Implementation (Insert, Traversals(Inorder, Preorder, Postorder), search)
Inorder:  [21, 23, 40, 56, 78]
Preorder:  [23, 21, 40, 56, 78]
Postorder:  [21, 40, 56, 78, 23]
Enter the value to be searched: 21
Element present
```

# PRACTICAL NO. 16

**Q. Linear Search Implementation in python.**

```
print("24167 - GAUTAM GANESH VELU")
print("Linear search python implementation")

def linear_search(arr,s):
  for i in range(len(arr)):
    if(arr[i]==s):
      return i
  return -1

a=[45,12,30,90]

n=int(input("Enter the element to search: "))
pos=linear_search(a,n)
if(pos==-1):
  print("Element not found")
else:
  print("\nElement found at position(index): ",pos)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Linear search python implementation
Enter the element to search: 30

Element found at position(index):  2
```

# PRACTICAL NO. 17

**Q. Binary search Implementation in python.**

```
print("24167 - GAUTAM GANESH VELU")
print("Binary search python implementation")

def binary_search(arr,s):
  lb=0
  ub=len(arr)-1

  while(lb<=ub):
    mid=(lb+ub)//2
    if(arr[mid]==s):
      return mid
    elif(s<arr[mid]):
      ub=mid-1
    else:
      lb=mid+1
  return -1

a=[3,5,7,10,20,36,45,58,90,100]
n=int(input("Enter the element to search: "))
pos=binary_search(a,n)
if(pos==-1):
  print("Element not found")
else:
  print("\nElement found at position(index): ",pos)
```

**OUTPUT:**

```
24167 - GAUTAM GANESH VELU
Binary search python implementation
Enter the element to search: 90

Element found at position(index):  8
```

# PRACTICAL NO. 18

**Q. Interpolation Search Implementation in python**

```
print("24167 - GAUTAM GANESH VELU")
print("Interpolation search python implementation")

def interpolation_search(arr,s):
  lb=0
  ub=len(arr)-1

  while(lb<=ub):
    mid=lb+ ((ub-lb)//(arr[ub]-arr[lb])*(s-arr[lb]))
    if(arr[mid]==s):
      return mid
    elif(s<arr[mid]):
      ub=mid-1
    else:
      lb=mid+1

  return -1
a=[3,5,7,10,20,36,45,58,90,100]
n=int(input("Enter the element to search: "))
pos=interpolation_search(a,n)
if(pos==-1):
  print("Element not found")
else:
  print("\nElement found at position(index): ",pos)
```

**OUTPUT:**

```
24167 - GAUTAM GANESH VELU
Interpolation search python implementation
Enter the element to search: 36

Element found at position(index):  5
```

# PRACTICAL NO. 19

**Q. Bubble Sort Implementation in python.**

```
print("24167 - GAUTAM GANESH VELU")
print("Bubble Sort python implementation")

def bubble_sort(arr):
   for i in range(len(arr)):
      for j in range(i+1,len(arr)):
         if(arr[i]>arr[j]):
            arr[i],arr[j]=arr[j],arr[i]

a=[56,12,23,90,33,8,59]
print("Unsorted array: ",a)
bubble_sort(a)
print(a)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Bubble Sort python implementation
Unsorted array:  [56, 12, 23, 90, 33, 8, 59]
[8, 12, 23, 33, 56, 59, 90]
```

# PRACTICAL NO. 20

**Q. Merge Sort Implementation in python.**

```
print("24167 - GAUTAM GANESH VELU")
print("Merge Sort python implementation")

def merge_sort(arr):
  if len(arr)<=1:
    return arr

  mid=len(arr)//2

  l_half=arr[:mid]
  r_half=arr[mid:]

  l_half=merge_sort(l_half)
  r_half=merge_sort(r_half)

  return merge(l_half,r_half)

def merge(left,right):
  new=[]
  i,j=0,0

  while i<len(left) and j<len(right):
    if left[i]<right[j]:
      new.append(left[i])
      i+=1
    else:
      new.append(right[j])
```

```
        j+=1
    new.extend(left[i:])
    new.extend(right[j:])
    return new

data=[45,23,12,78,90,22,8,56]
print("Unsorted list is: ", data)
sorted_data=merge_sort(data)

print("\nSorted list is: ")
print(sorted_data)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Quick Sort python implementation
[11, 12, 22, 32, 45, 77, 78, 90]
```

# PRACTICAL NO. 21

**Q. Quick Sort Implementation in python.**

```
print("24167 - GAUTAM GANESH VELU")
print("Quick Sort python implementation")


def quick_sort(arr,low,high):
    if low<high:
        pivot=partition(arr,low,high)
        quick_sort(arr,low,pivot-1)
        quick_sort(arr,pivot+1,high)


def partition(arr,low,high):

    p=arr[low]
    i=low+1
    j=high

    while True:
        while i<=j and arr[i]<=p:
            i+=1
        while i<=j and arr[j]>=p:
            j-=1

        if i<=j:
            arr[i],arr[j]=arr[j],arr[i]
        else:
            break

    arr[low],arr[j]=arr[j],arr[low]
```

```
    return j


data=[32,78,45,12,90,22,77,11]
quick_sort(data,0,7)
print(data)
```

## OUTPUT:

```
24167 - GAUTAM GANESH VELU
Merge Sort python implementation
Unsorted list is:  [45, 23, 12, 78, 90, 22, 8, 56]

Sorted list is:
[8, 12, 22, 23, 45, 56, 78, 90]
```