****

**Data Structures and Algorithm**

**Practical Journal**

INDEX

|  |  |  |
| --- | --- | --- |
| Sr. No. | PRACTICALS | REMARKS |
|  | **Create Arrays of different data types in python using array module** |  |
|  | **Implement different possible operations on Array in python using array module** |  |
|  | **Create Matrix using Numpy module & do matrix operations (addition, Subtraction, Multiplication & Transpose) in python.** |  |
|  | **Create sparse matrix & display its transpose in python** |  |
|  | **Implement Singly Linked List data structure in python** |  |
|  | **Implement Doubly Linked List data structure in python** |  |
|  | **Implement Singly Circular Linked List data structure in python** |  |
|  | **Write a program to reverse a linked list in python** |  |
|  | **Implement Stack Static data structure in python** |  |
|  | **Implement Stack Dynamic data structure in python** |  |
|  | **Static implementation of queue data structure in python using list** |  |
|  | **Python program to implement a Queue using singly linked list [Dynamic Implementation]** |  |
|  | **Python program to reverse a string using stack** |  |
|  | **Python program to evaluate postfix expression using stack** |  |
|  | **BST (Binary Search tree) Implementation (Insert, Traversals**  **(Inorder, Preorder, Postorder), search)** |  |
|  | **Linear Search Implementation in python** |  |
|  | **Binary search Implementation in python** |  |
|  | **Interpolation Search Implementation in python** |  |
|  | **Bubble Sort Implementation in python** |  |
|  | **Merge Sort Implementation in python** |  |
|  | **Quick Sort Implementation in python** |  |

**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")

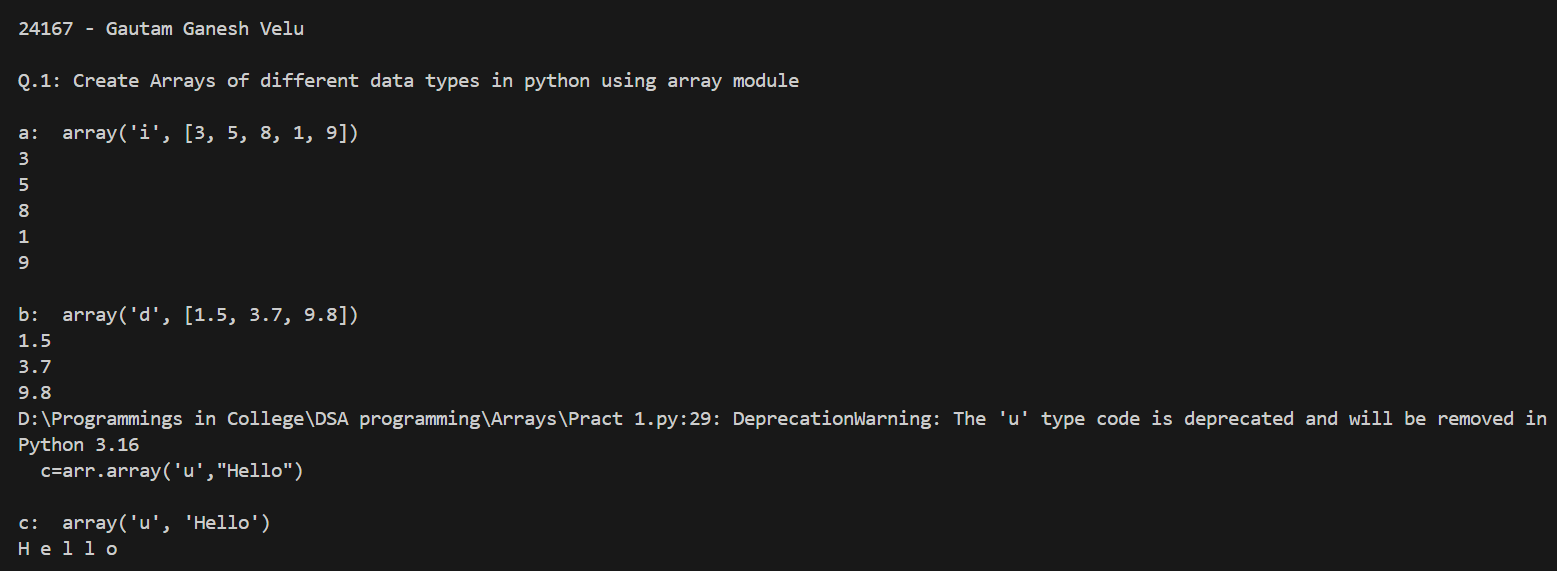
print("\nc: ",c)

#traversing

for x in c:

    print(x,end=' ')

**OUTPUT:**



**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)

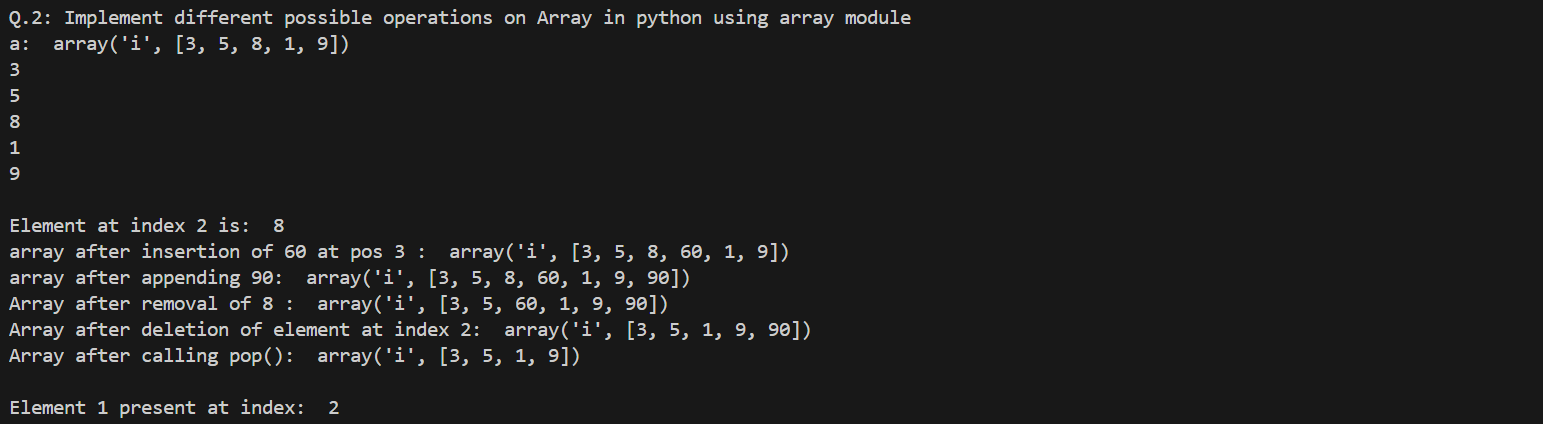
#Search

#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:**



**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

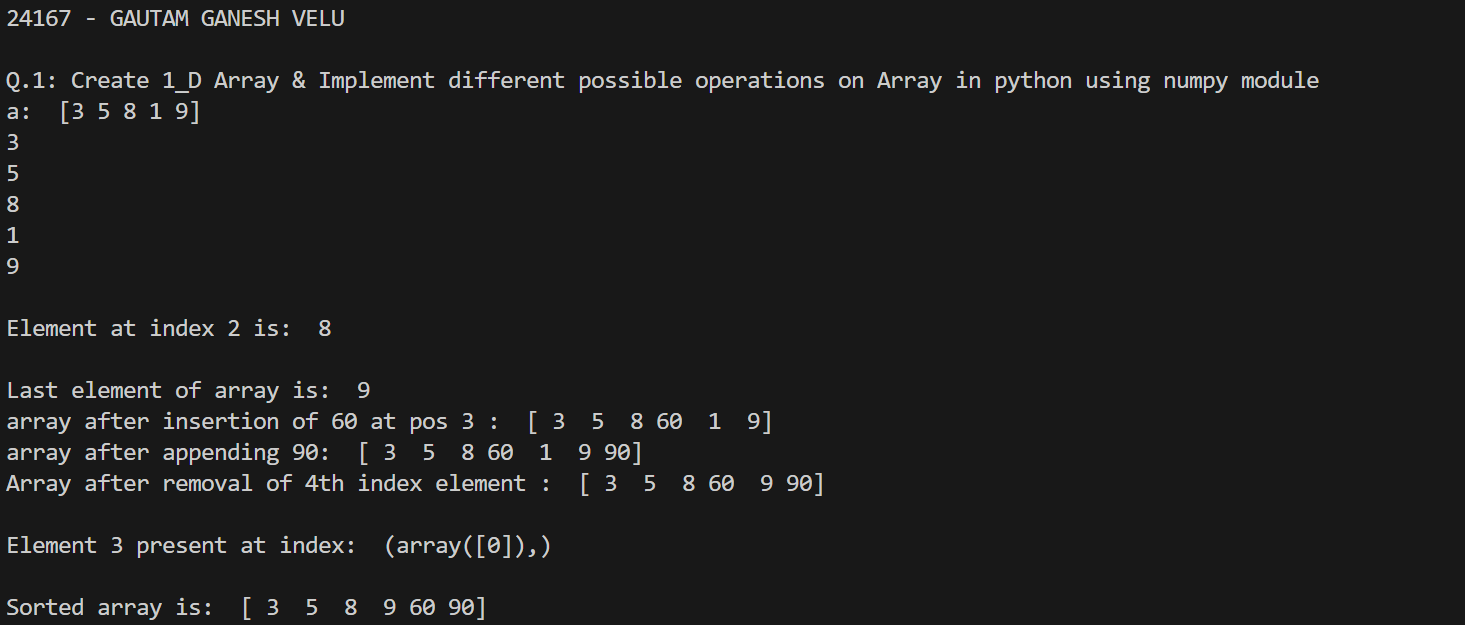
#where(ele) method can be used to search the given ele if it is present then it will return the index or it will generate error

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

#Sorting

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

**OUTPUT:**



**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

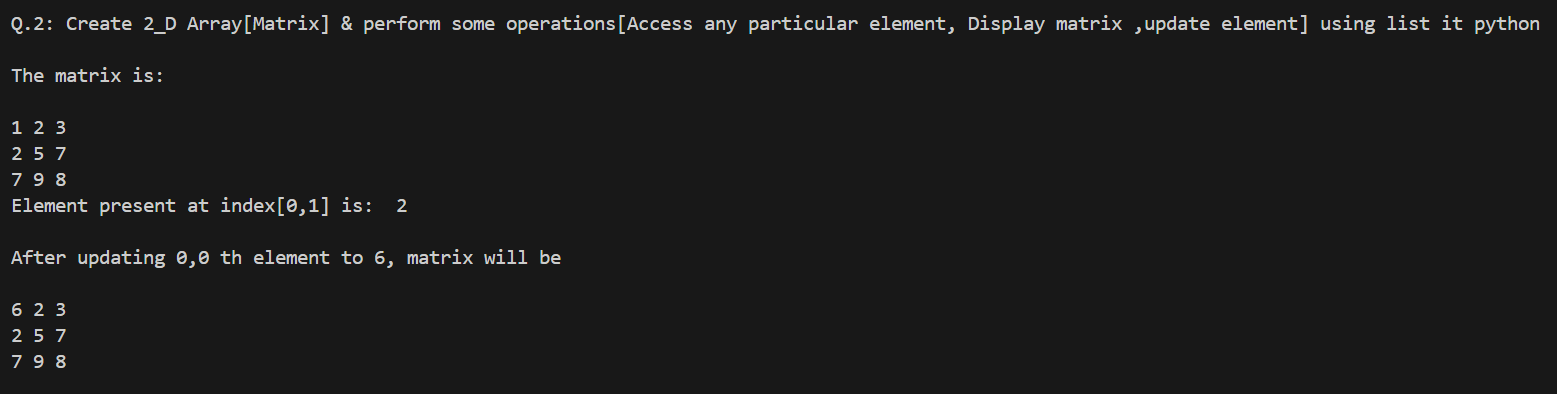
print("\nAfter updating 0,0 th element to 6, matrix will be\n")

for i in range(len(mat1)):

    for j in range(len(mat1[0])):

        print(mat1[i][j],end=' ')

    print()

**OUTPUT:** 

**Q 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: ")

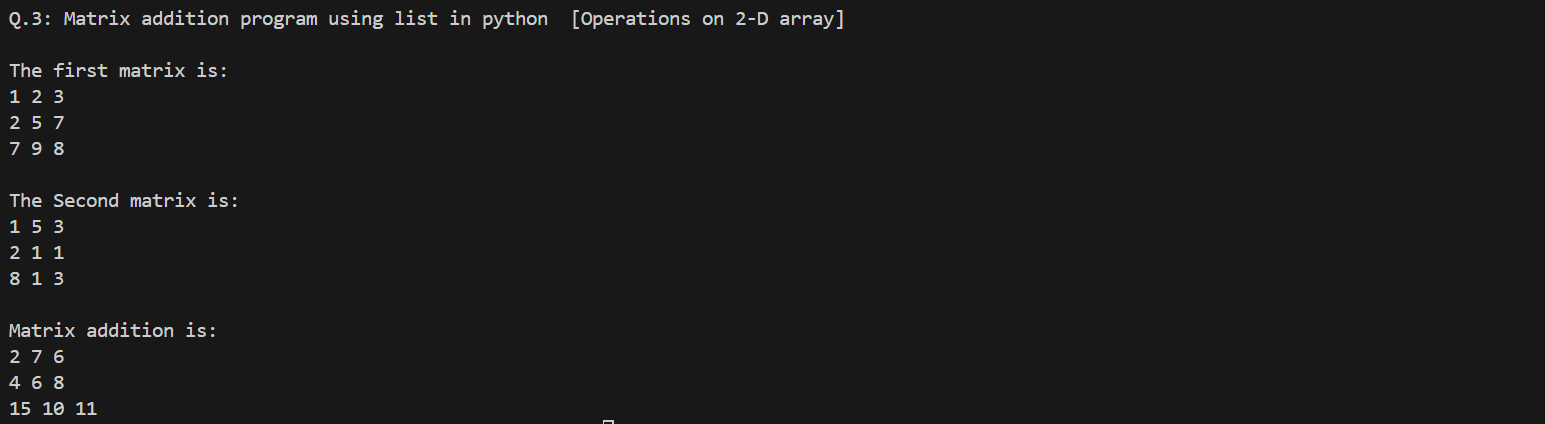
for i in range(len(mat3)):

    for j in range(len(mat3[0])):

        print(mat3[i][j],end=' ')

    print()

**OUTPUT:**



**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)

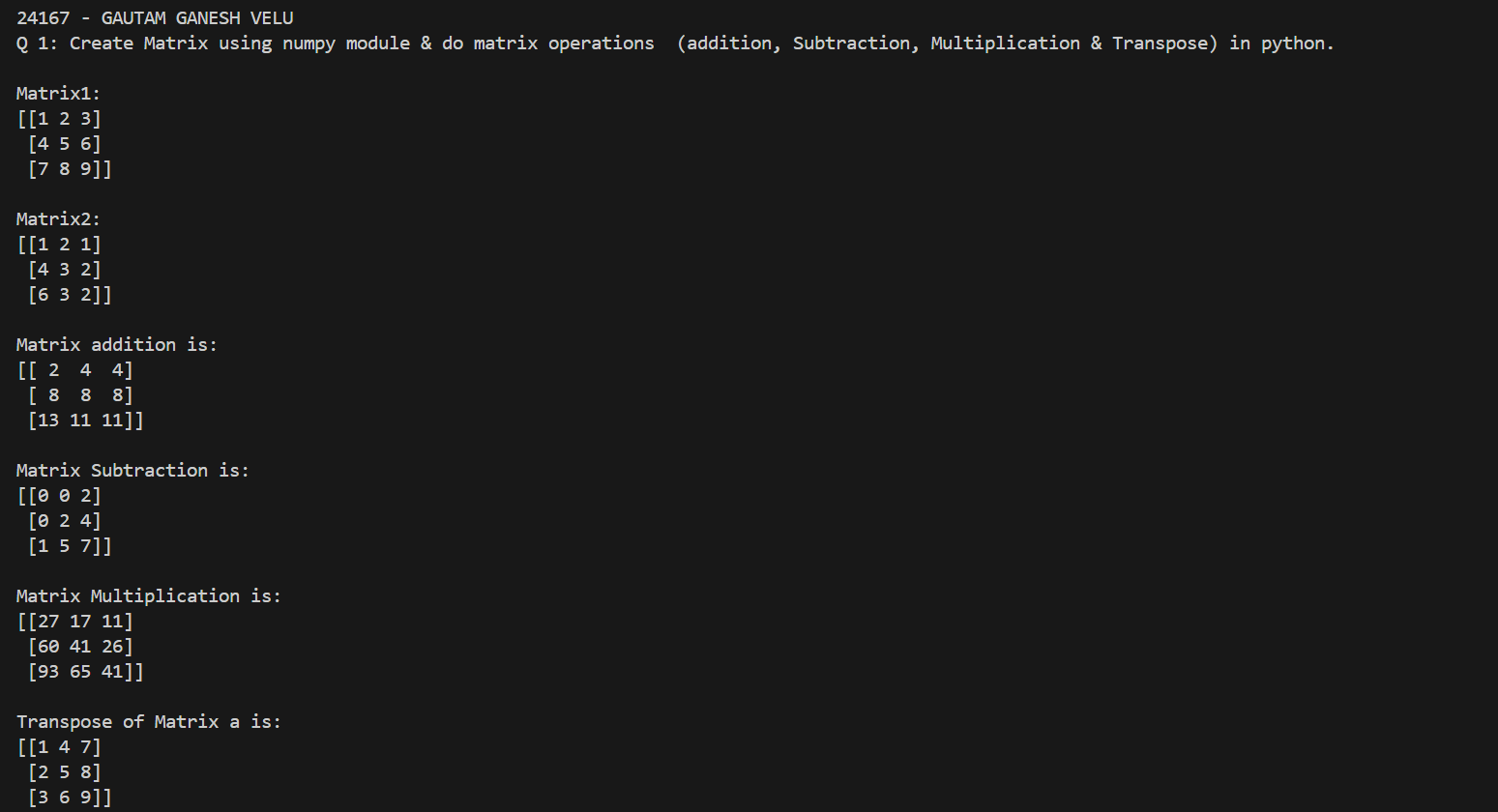
#matrix transpose

t=a.transpose()

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

print(t)

**OUTPUT:**



**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:**

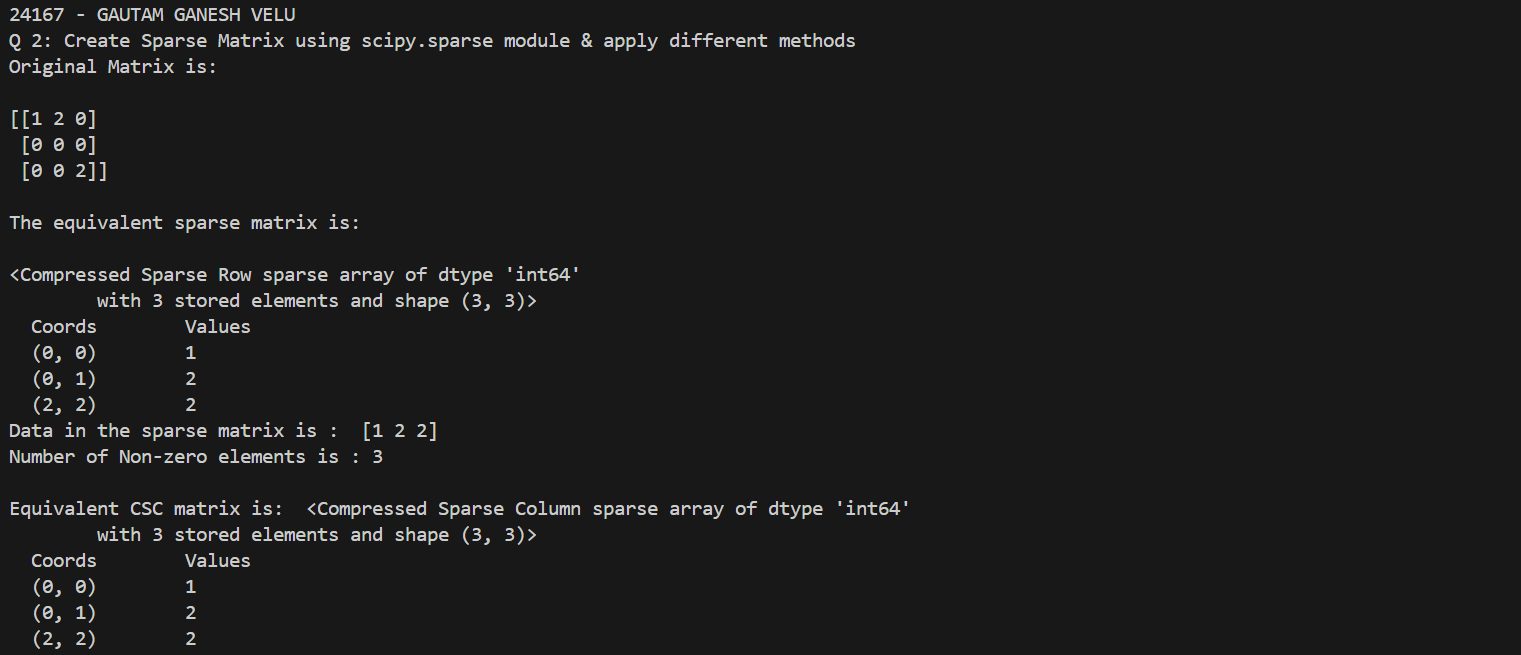
**print("Number of Non-zero elements is :",SA.count\_nonzero())**

**#Converting from csr to csc with the tocsc() method:**

**new\_mat=SA.tocsc()**

**print("\nEquivalent CSC matrix is: ",new\_mat)**

**OUTPUT:**



**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)**

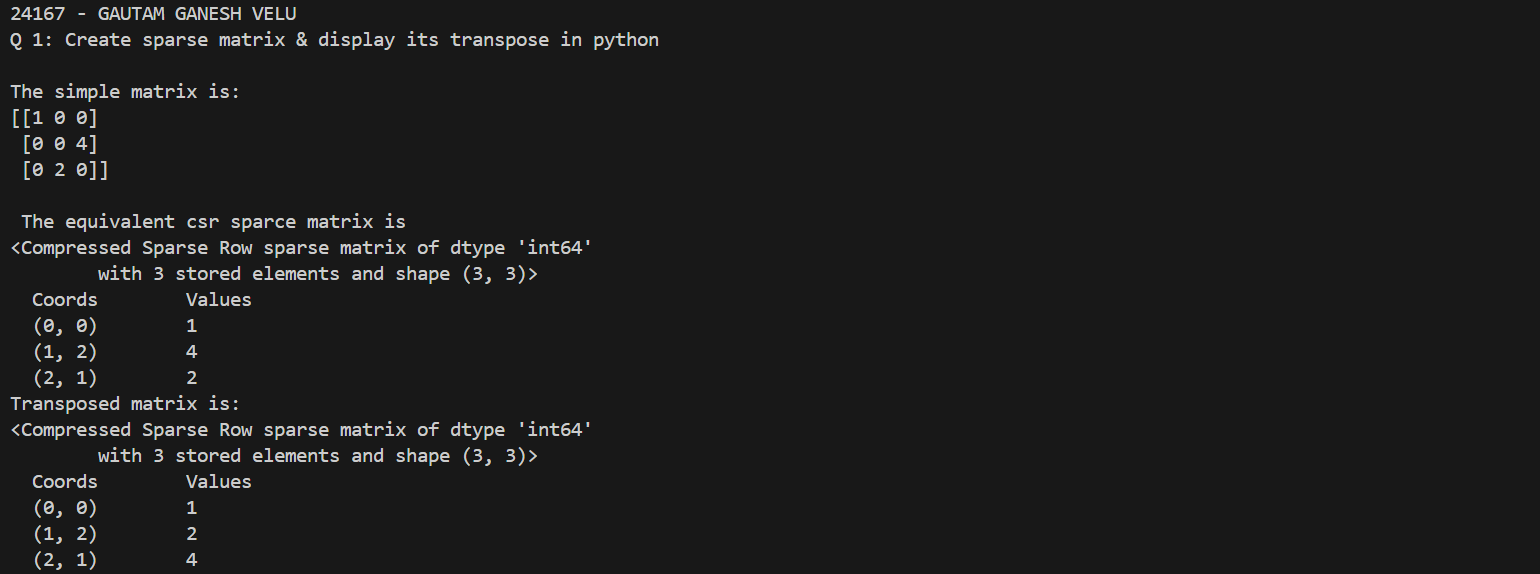
**#sparse matrix transpose**

**b=sm.transpose().tocsr()**

**print("Transposed matrix is: ")**

**print(b)**

**OUTPUT:**



**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)**

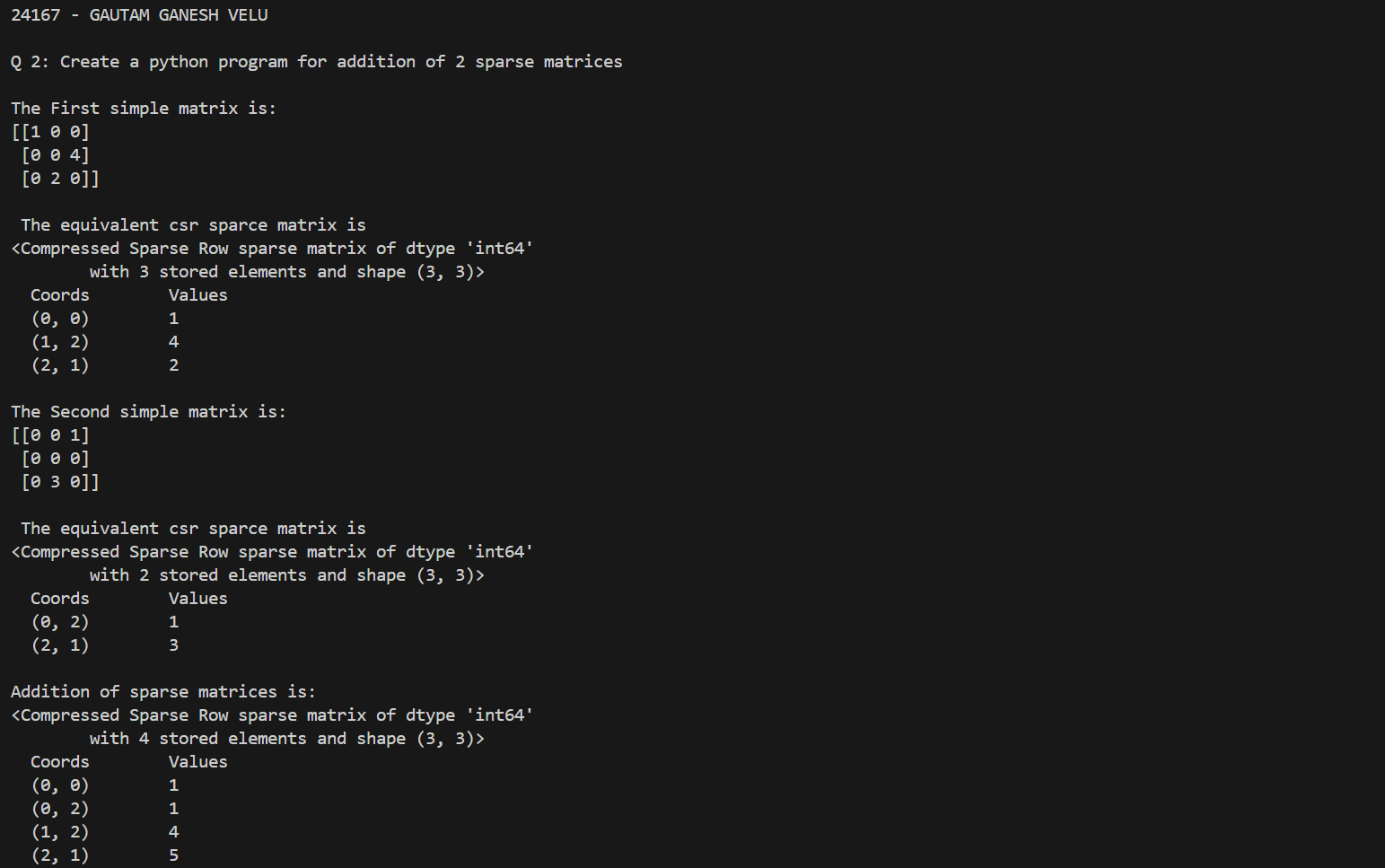
**#addition**

**c=sm1+sm2**

**print("\nAddition of sparse matrices is: ")**

**print(c)**

**OUTPUT:**



**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)**

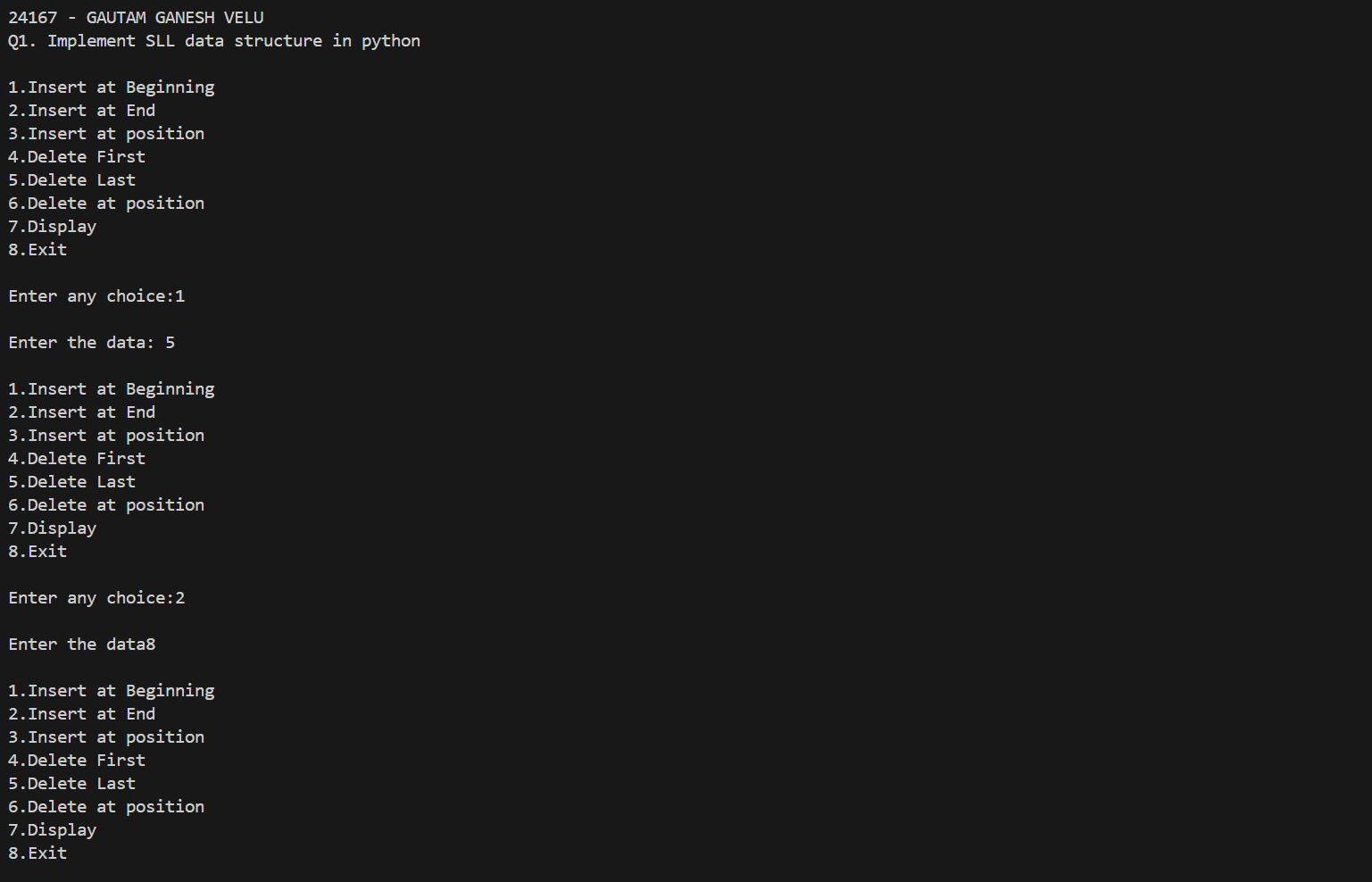
**elif(ch==7):**

**l.display()**

**else:**

**print("Invalid Option!!")**

**OUTPUT:**

****

**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)**

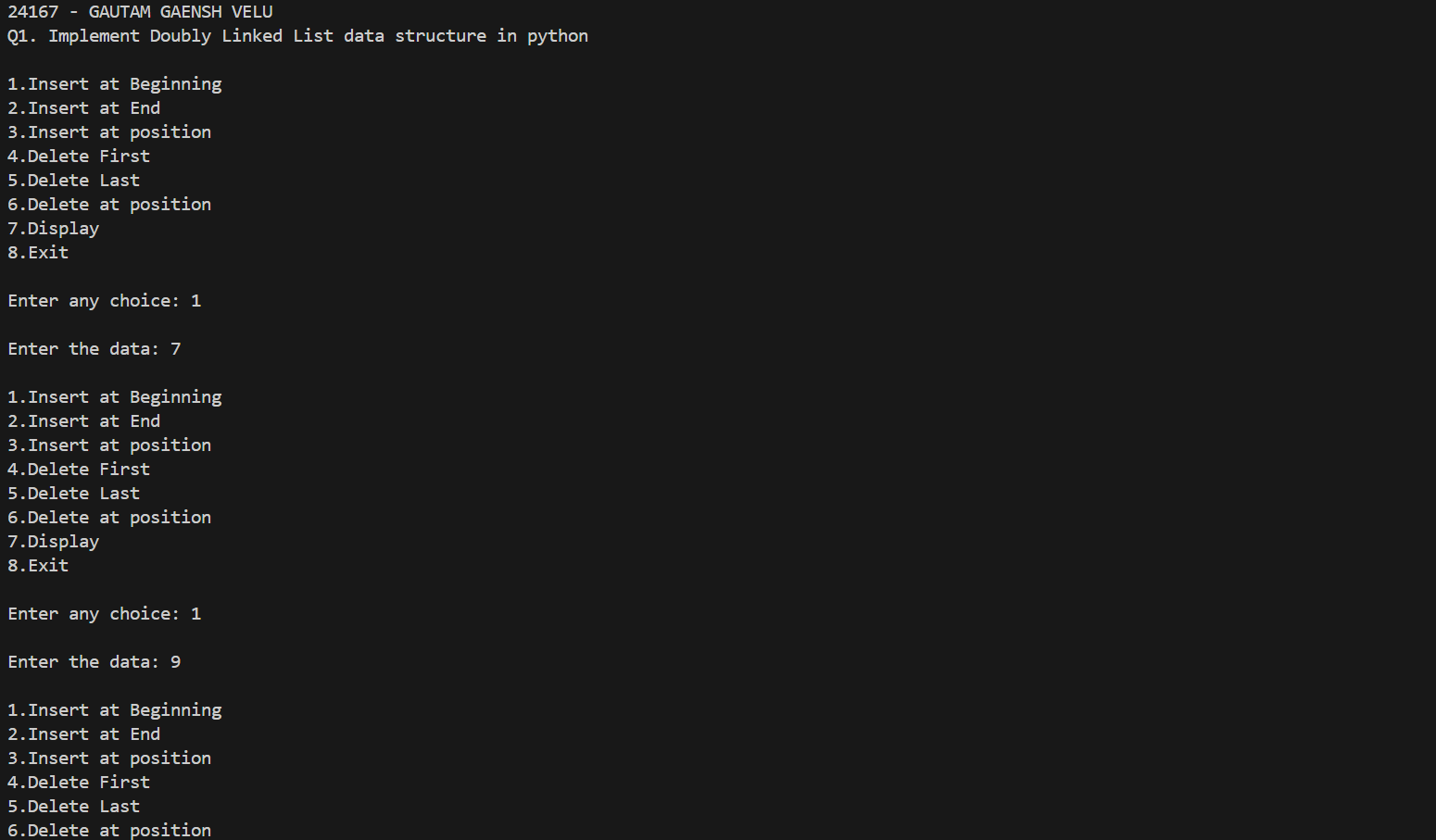
**elif(ch==7):**

**l.display()**

**else:**

**print("Invalid option!!")**

**OUTPUT:**





**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)**

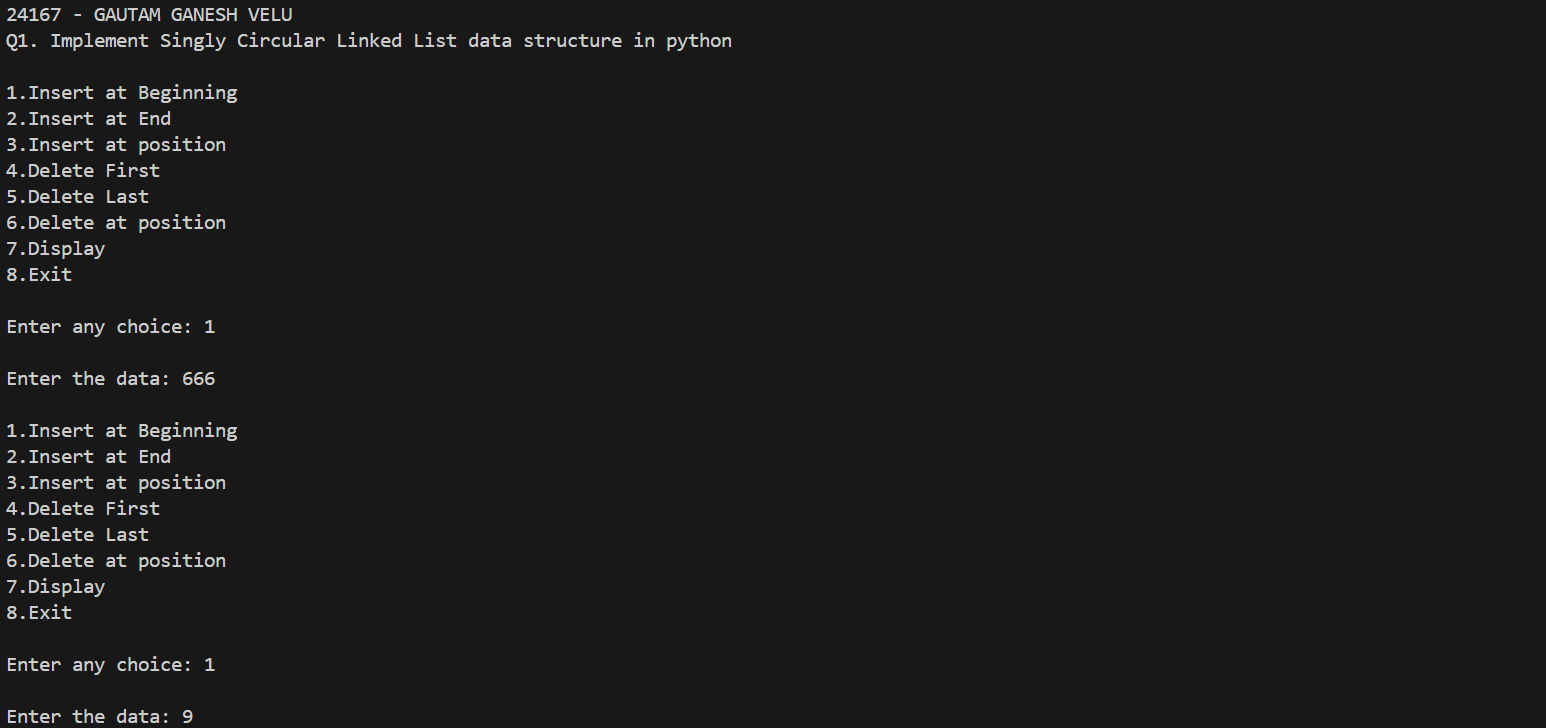
**elif(ch==7):**

**l.display()**

**else:**

**print("Invalid option!!")**

**OUTPUT:**

**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: ")**

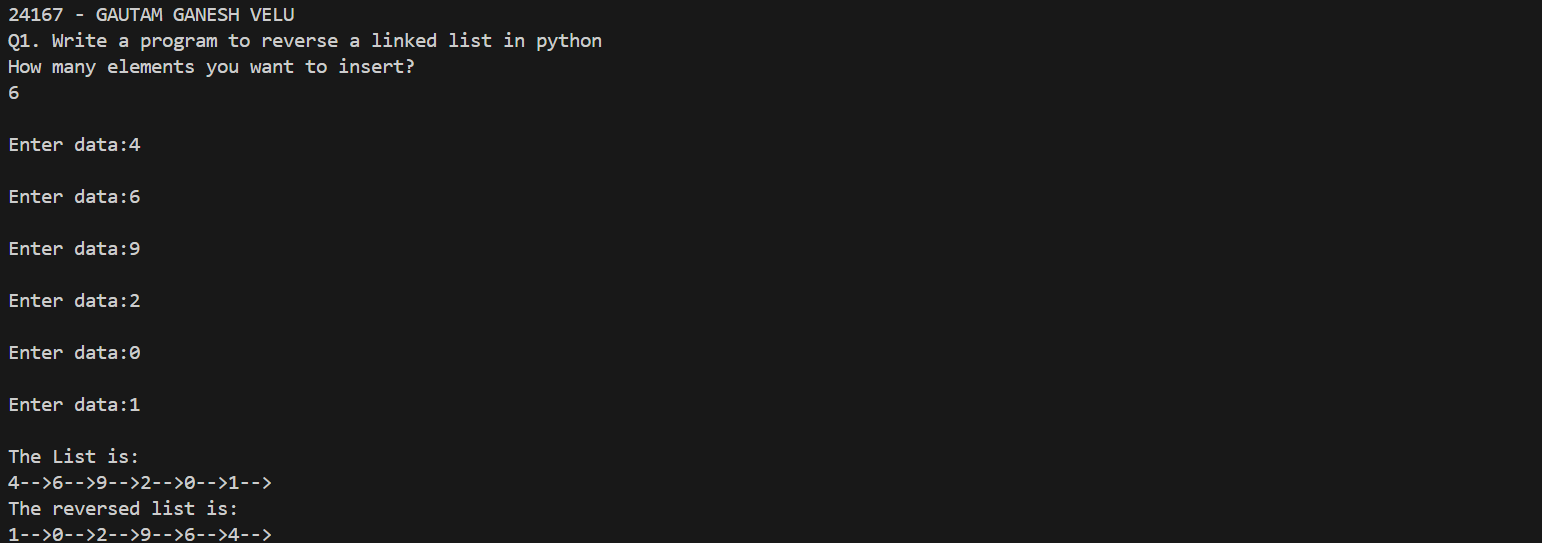
**L.display()**

**print("\nThe reversed list is: ")**

**L.reverse()**

**L.display()**

**OUTPUT:**



**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())**

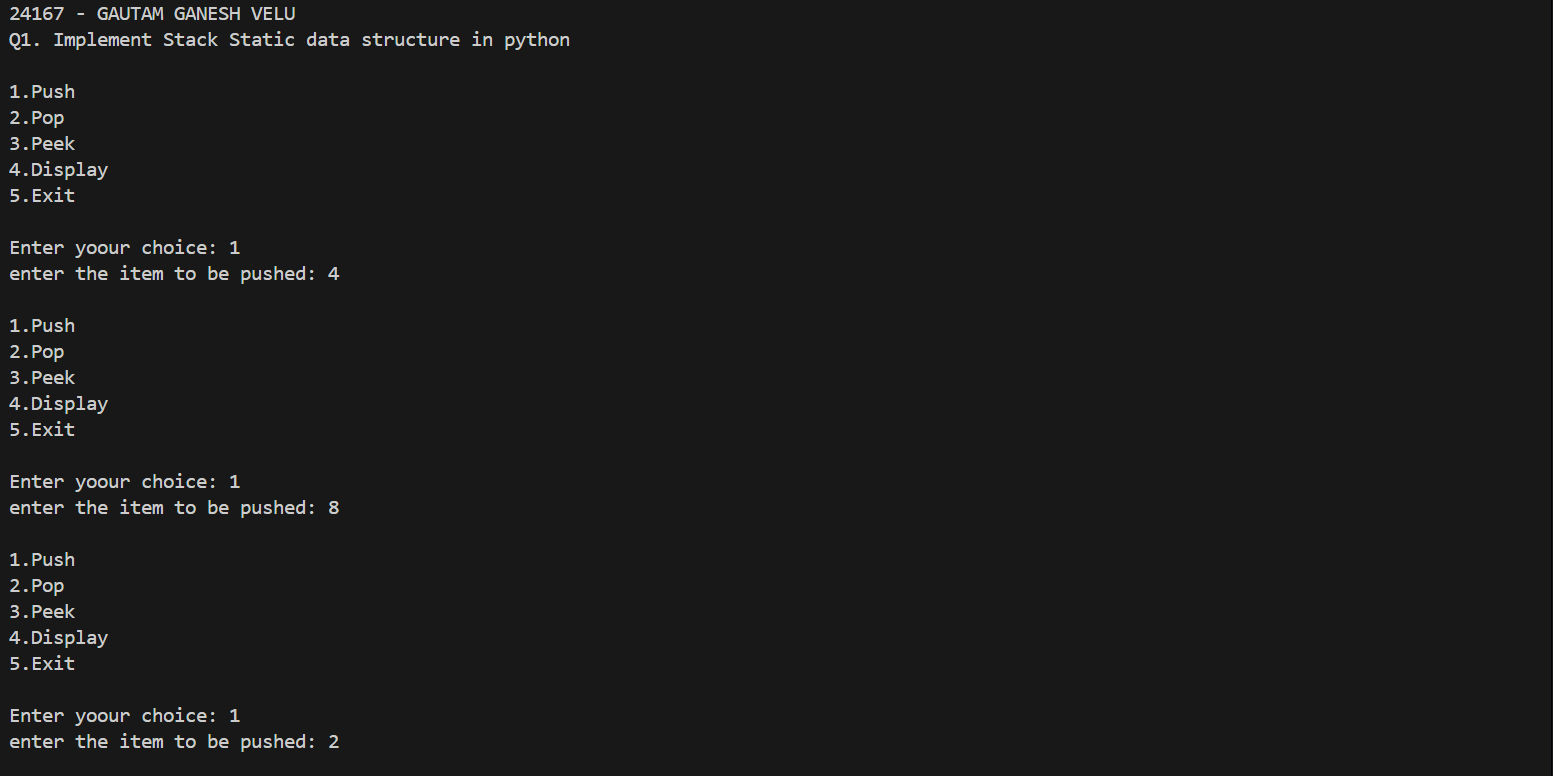
**elif(ch==3):**

**print("Peeked element is: ",s.peek())**

**else:**

**s.display()**

**OUTPUT:**

**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()**

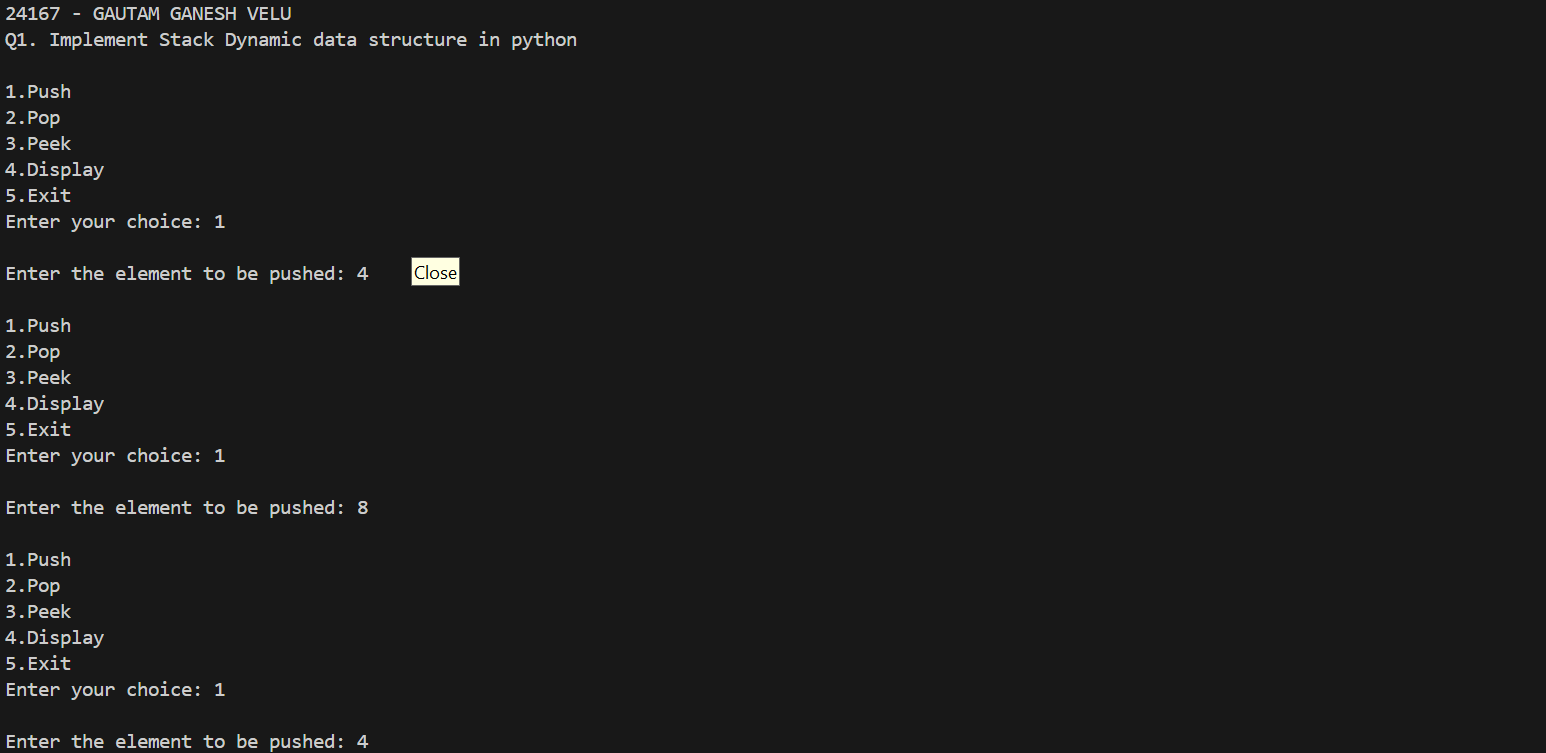
**elif ch == 3:**

**st.peek()**

**else:**

**st.display()**

**OUTPUT:**

**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)**

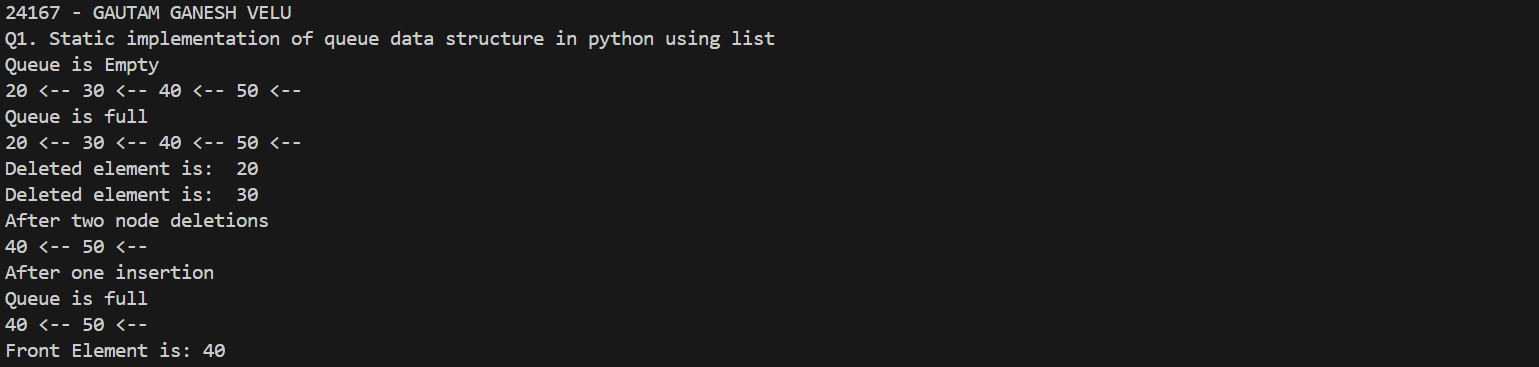
**# Print queue elements**

**q.display()**

**# Print front of the queue**

**q.front\_element()**

**OUTPUT:**



**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)**

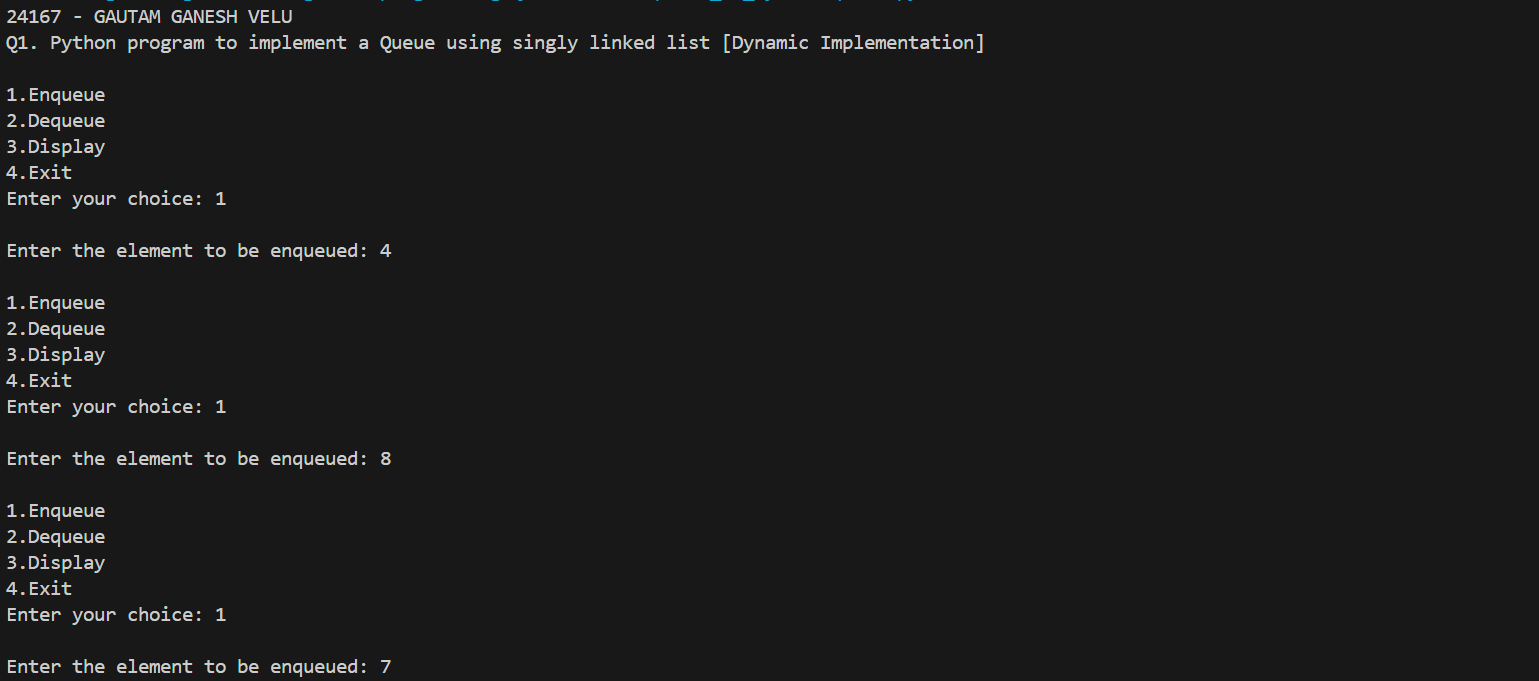
**elif ch==2:**

**Q.dequeue()**

**else:**

**Q.display()**

**OUTPUT:**

**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=[]**

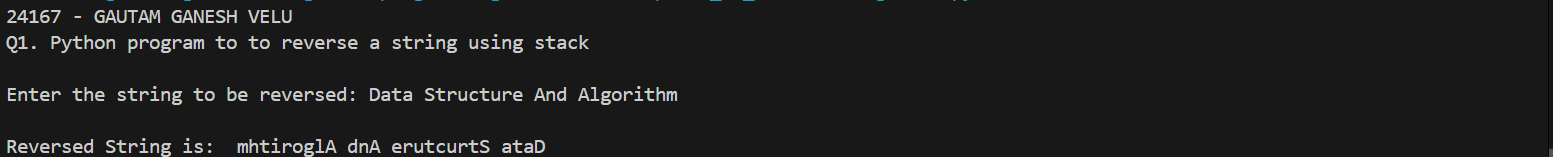
**while (not st.is\_empty()):**

**rlist.append(st.pop())**

**rstr=''.join(rlist)**

**print("\nReversed String is: ",rstr)**

**OUTPUT:**



**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:**



**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: "))**

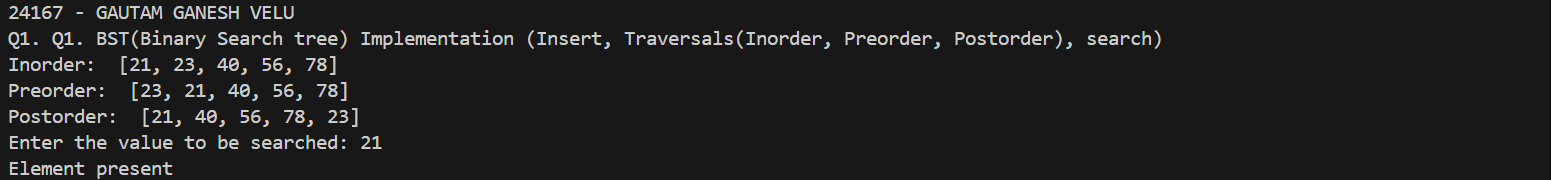
**if(b.search(s)):**

**print("Element present")**

**else:**

**print("Element Not present")**

**OUTPUT:**



**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)**

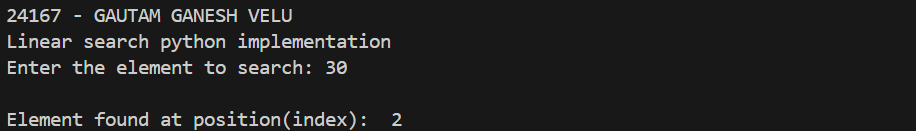
**if(pos==-1):**

**print("Element not found")**

**else:**

**print("\nElement found at position(index): ",pos)**

**OUTPUT:**



**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:**

****

**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)**

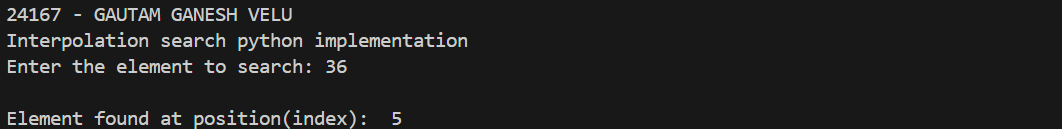
**if(pos==-1):**

**print("Element not found")**

**else:**

**print("\nElement found at position(index): ",pos)**

**OUTPUT:**

****

**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:**

****

**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:**

****

**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:**

****