1. Python program to Use and demonstrate basic data structures.

```
print("List")
l1 = [1, 2,"ABC", 3, "xyz", 2.3]
print(l1)
print("Dictionary")
d1={"a":134,"b":266,"c":343}
print(d1)
print("Tuples")
t1=(10,20,30,40,50,40)
print (t1)
print("Sets")
s1={10,30,20,40,10,30,40,20,50,50}
print(s1)
```

2. Implement an ADT with all its operations.

```
class Date:
    def_init_(self,d,m,y):
        self.d=d
       self.m=m
       self.y=y
    def  day(self):
       print("Day = ", self.d)
    def month(self):
       print("Month = ", self.m)
    def year(self):
       print("year = ", self.y)
    def monthName(self):
       months = ["Unknown","January","Febuary","March","April","May","June","July",
        "August","September","October","November","December"]
       print("Month Name:",months[self.m])
    def isLeapYear(self):
       if (self.y % 400 == 0) and (self.y % 100 == 0):
          print("It is a Leap year")
       elif (self.y % 4 == 0) and (self.y % 100 != 0):
         print("It is a Leap year")
       else:
         print("It is not a Leap year")


dd=int(input("Enter the day:"))
mm=int(input("Enter the month:"))
yy=int(input("Enter the year:"))
```

```
d1 = Date(dd,mm,yy)
d1.day()
d1.month()
d1.year()
d1.monthName()
d1.isLeapYear()
```

3. Implement an ADT and Compute space and time complexities.

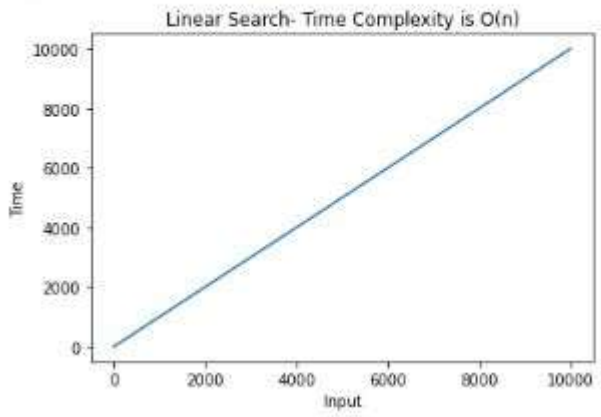
```
import time
class Stack:
    def__init_(self):
        self.items = []
    def  isEmpty(self):
        return self.items == []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def peek(self):
        return self.items[len(self.items) - 1]
    def size(self):
        return len(self.items)
    def display(self):
        return (self.items)
s=Stack()
start = time.time()
print(s.isEmpty())
print("push operations")
s.push(11)
s.push(12)
s.push(13)
print("size:",s.size())
print(s.display())
print("peek",s.peek())
print("pop operations")
print(s.pop())
print(s.pop())
print(s.display())
print("size:",s.size())
end = time.time()
print("Runtime of the program is", end - start)
```

4. Implement Linear Search and compute space and time complexities, plot graph using asymptomatic notations

```
import time
import matplotlib.pyplot as plt
#Function Definition
def linearsearch(a, key):
    n = len(a)
    for i in range(n):
        if a[i] == key:
            return i;
    return -1
# code to call the function linearsearch()
a = [13,24,35,46,57,68,79]
start = time.time()
print(f "the array elements are:{a}")
key = int(input("enter the key element to search:"))
result = linearsearch(a,key)
if result == -1:
    print("Search UnSuccessful")
else:
    print("Search Successful key found at  %d location:" %result)
end = time.time()
print("Runtime of the program is", end-start)

x=list(range(1,10000))
plt.plot(x , [y for y in x] )
plt.title("Linear Search- Time Complexity is O(n)")
plt.xlabel("Input")
plt.ylabel("Time")
plt.show()
```

Output:

```
the array elements are: [13, 24, 35, 46, 57, 68, 79]
enter the key element to search:12
Search UnSuccessful
Runtime of the program is 2.6451847553253174
```



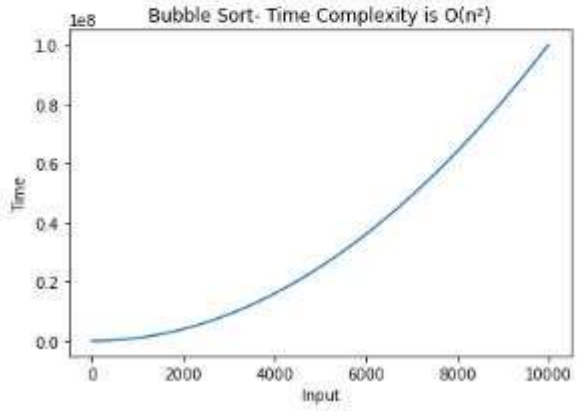
5. Implement Bubble Sort and compute space and time complexities, plot graph using asymptomatic notations

#program to implement bubble sort

```
import time
import matplotlib.pyplot as plt
# bubblesort function definition
def bubblesort(list1):
   n=len(list1)
   for i in range(n-1): # for loop to represent the number of steps
        # swapped flag is set to false in beginning of every step to check whether the items are already sorted
        swapped=False
        for j in range(n-1-i): #for loop to represent the number of comparisons in every step
                if list1[j]>list1[j+1]: # compare the adjacent elements
                        list1[j],list1[j+1]=list1[j+1],list1[j] # exachange only if they are in wrong order
                        swapped=True
        if swapped==False:
         break
   return list1
list1=[]
n=int(input("How many elements??"))
for i in range(n):
   list1.append(int(input("Enter %d number:" %i)))

print(f"before swapping:{list1}")

list1=bubblesort(list1)
print(f'After swapping:{list1}')

x=list(range(1,10000))
plt.plot(x , [y*y for y in x] )
plt.title("Bubble Sort- Time Complexity is O(n\u00b2)")
plt.xlabel("Input")
plt.ylabel("Time")
plt.show()
```

Output:

```
How many elements??5
Enter 0 number:12
Enter 1 number:34
Enter 2 number:21
Enter 3 number:67
Enter 4 number:4
before swapping:[12, 34, 21, 67, 4]
After swapping:[4, 12, 21, 34, 67]
```

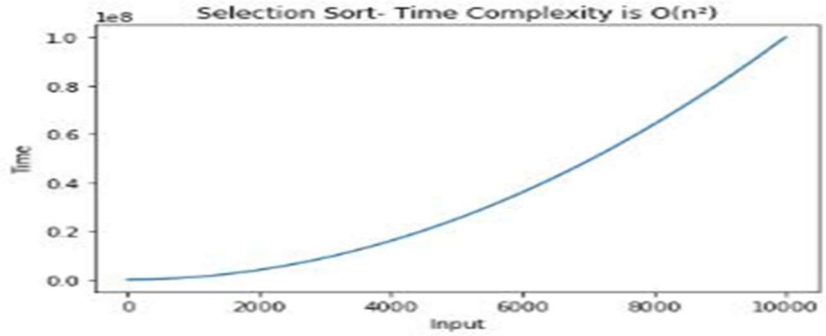


6. Implement Selection Sort and compute space and time complexities, plot graph using asymptomatic notations

```
import matplotlib.pyplot as plt
def selectionsort(array):
        n=len(array)
        for i in range(n-1):
            min=i
            for j in range(i+1,n):
                if array[j]<array[min]:
                    min=j
            array[i],array[min]=array[min],array[i]
        return array
array=[]
n=int(input("How many elements??"))
for i in range(n):
    array.append(int(input("Enter %d number:" %i)))
print(f"before swapping:{array}")
array=selectionsort(array)
print(f'After swapping:{array}')
x=list(range(1,10000))
plt.plot(x , [y*y for y in x] )
plt.title("Selection Sort- Time Complexity is O(n\u00b2)")
plt.xlabel("Input")
plt.ylabel("Time")
plt.show()
```

OUTPUT:-

```
How many elements??10
Enter 0 number:34
Enter 1 number:23
Enter 2 number:89
Enter 3 number:-5
Enter 4 number:21
Enter 5 number:90
Enter 6 number:57
Enter 7 number:45
Enter 8 number:31
Enter 9 number:-78
before swapping:[34, 23, 89, -5, 21, 90, 57, 45, 31, -78]
After swapping:[-78, -5, 21, 23, 31, 34, 45, 57, 89, 90]
```

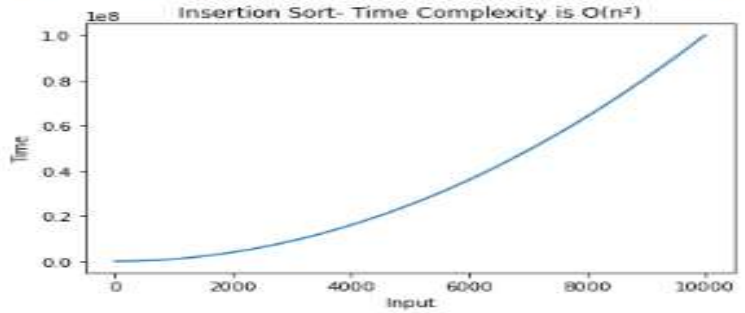


7. Implement Insertion Sort and compute space and time complexities, plot graph using asymptomatic notations

```
import matplotlib.pyplot as plt
def insertionsort(array):
    n=len(array)
    for step in range(1,n):
        item=array[step]
        j=step-1
        while j>=0 and item<array[j]:
            array[j+1]=array[j]
            j=j-1
        array[j+1]=item
    return array

array=[]
n=int(input("How many elements??"))
for i in range(n):
    array.append(int(input("Enter %d number:" %i)))
print(f"before swapping:{array}")
array=insertionsort(array)
print(f'After swapping:{array}')
x=list(range(1,10000))
plt.plot(x , [y*y for y in x] )
plt.title("Insertion Sort- Time Complexity is O(n\u00b2)")
plt.xlabel("Input")
plt.ylabel("Time")
plt.show()
```

OUTPUT

```
How many elements??6
Enter 0 number:21
Enter 1 number:34
Enter 2 number:12
Enter 3 number:45
Enter 4 number:-34
Enter 5 number:56
before swapping:[21, 34, 12, 45, -34, 56]
After swapping:[-34, 12, 21, 34, 45, 56]
```



SECTION-II

8. Implement Binary Search using Recursion and compute space and time complexities, plot graph using asymptomatic notations

```
import time
def binarysearch(a, low, high, key):
    if low <= high:
        mid = (high + low) // 2
        if a[mid] == key:
            print("Search Successful key found at location:",mid)
            return
        elif key < a[mid]:
            binarysearch(a, low, mid-1, key)
        else :
            binarysearch(a, mid + 1, high, key)
    else:
        print("Unsuccessful Search")

a=[ ]
n=int(input("How many elements??:"))
for i in range(n):
    a.append(int(input("Enter the number")))
print("the array elements are:",a)
key = int(input("enter the key element to search:"))
start=time.time()
binarysearch(a,0,len(a)-1,key)
end=time.time()
print("The time taken for binary search is",end-start)
```

OUTPUT:

```
How many elements??:5
Enter the number23
Enter the number45
Enter the number78
Enter the number89
Enter the number99
the array elements are: [23, 45, 78, 89, 99]
enter the key element to search:34
Unsuccessful Search
The time taken for binary search is 0.0
```

9. Implement Merge Sort and compute space and time complexities, plot graph using asymptomatic notations

```
# Function definition for Mergesort
Import time
def mergesort(list1):
    if len(list1)>1:
        mid=len(list1)//2 # Divide list into 2 halves
        left=list1[:mid]
        right=list1[mid:]
        mergesort(left)
        mergesort(right)
        i=j=k=0
        while i<len(left) and j<len(right):
            if left[i]<right[j]:
                list1[k]=left[i]
                i+=1
            else:
                list1[k]=right[j]
                j+=1
            k+=1
        while i<len(left):
            list1[k]=left[i]
            i+=1
            k+=1
        while j<len(right):
            list1[k]=right[j]
            j+=1
            k+=1
    return list1

list1=[]
n=int(input("Enter the size of  list"))
for i in range(n):
    list1.append(int(input("Enter the number")))
print("Before sorting: The list items are")
for i in range(len(list1)):
    print(list1[i],end=" ")
start=time.time()
```

```
list1=mergesort(list1)
end=time.time()
print()
for i in range(len(list1)):
    print(list1[i],end=" ")
```

Output:

```
Enter the size of  list5
Enter the number23
Enter the number12
Enter the number34
Enter the number56
Enter the number78
Before sorting: The list items are
23 12 34 56 78
After sorting: The list items are

12 23 34 56 78
The time taken for merge sort is  0.0
```

10. Implement Quick Sort and compute space and time complexities, plot graph using asymptomatic notations

```
import time
def partition(array,start,end):
   pivot=array[start]
   low=start+1
   high=end
   while True:
      while low<=high and array[low]<pivot:
         low+=1
      while low<=high and array[high]>=pivot:
         high=high-1
      if low<=high:
        array[low],array[high]=array[high],array[low]
      else:
         break
  array[start],array[high]=array[high],array[start]
  return high

#Code for the Quicksort() which divides the
array into two halves
def Quicksort(array,start,end):
   if start>=end:
      return
   p=partition(array,start,end)
   Quicksort(array,start,p-1)
   Quicksort(array,p+1,end)

#driver code to call Quicksort function
array=[]
n=int(input("Enter the size of  list:"))
for i in range(n):
   array.append(int(input("Enter the %d number:" %i)))
print("Before sorting: The list items are")
for i in range(len(array)):
   print(array[i],end=" ")
start=time.time()
Quicksort(array,0,len(array)-1)
```

```
end=time.time()
print("\nAfter sorting: The list items",array)
```

OUTPUT:

```
Enter the size of  list:5
Enter the 0 number:12
Enter the 1 number:23
Enter the 2 number:11
Enter the 3 number:56
Enter the 4 number:22
Before sorting: The list items are
12 23 11 56 22
After sorting: The list items
[11, 12, 22, 23, 56]
The time taken is  0.0
```

11. Implement Fibonacci sequence with dynamic programming

```
#function definition for fib()
def fib(n):
    if n <= 1:
        return n
    f = [0, 1]
    for i in range(2, n + 1):
        f.append(f[i - 1] + f[i - 2])
    print("The Fibonacci sequence is:", f)

#driver code to call the function fib()
n = int(input("Enter the term:"))
fib(n)
```

OUTPUT:

```
Enter the term:8
The Fibonacci sequence is: [0, 1, 1, 2, 3, 5, 8, 13, 21]
```

12. Implement singly linked list (Traversing the Nodes, searching for a Node, Prepending Nodes, and Removing Nodes)

```
#Class to create structure of node in a singly linked list
class Node:
    def __init__(self, data = None):
        self.data = data
        self.next = None
#create the data structure SinglyLinkedList
class SinglyLinkedList:
    def __init__(self):
        self.first = None

    def insertFirst(self, data):
      newnode = Node(data)
      if self.first==None:
          self.first =newnode
      else:
          newnode.next=self.first
          self.first=newnode

    def removeFirst(self):
        if(self.first== None):
            print("list is empty")
        else:
            cur=self.first
            self.first=self.first.next
            print("the deleted item is",cur.data)

    def display(self):
        if(self.first== None):
            print("list is empty")
            return
        cur = self.first
        while(cur):
            print(cur.data, end = " ")
            cur = cur.next
```

```
    def search(self,item):
        if(self.first== None):
            print("list is empty")
            return
        cur = self.first
        found=False
        while cur != None and not found:
            if cur.data == item:
                found=True
            else:
                cur=cur.next
        if found:
            print("The data item is present in the list")
        else:
            print("The Data item is not present")

#Singly Linked List
sll = SinglyLinkedList()
while(True):
    choice = int(input("\nEnter your choice 1-insert 2-delete 3-search 4-display 5-exit :"))
    if ( choice == 1):
        item = input("Enter the element to insert:")
        sll.insertFirst(item)
        sll.display()
    elif ( choice == 2):
        sll.removeFirst()
        sll.display()
    elif ( choice == 3):
        item = input("Enter the element to search:")
        sll.search(item)
    elif ( choice == 4):
        sll.display()
    else:
        break
```

OUTPUT:

```
Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :2
list is empty
list is empty

Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :3
Enter the element to search:23
list is empty

Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :4
list is empty

Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :1
Enter the element to insert:GPT Athani
GPT Athani
Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :1
Enter the element to insert:179
179 GPT Athani
Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :1
Enter the element to insert: DTE
 DTE 179 GPT Athani
Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :3
Enter the element to search:179
The data item is present in the list

Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :3
Enter the element to search:45.6
The Data item is not present

Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :4
 DTE 179 GPT Athani
Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :2
the deleted item is  DTE
179 GPT Athani
Enter your choice 1-insert 2-delete 3-search 4-display 5-exit :2
the deleted item is 179
GPT Athani
```

13. Implement linked list Iterators

```
class Node:
    def __init__(self, data = None):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.first = None
    #Function to insert the node at the end of list
    def insert(self, data):
        newnode = Node(data)
        if self.first == None:
            self.first = newnode
        else:
            cur=self.first
            while(cur.next):
                cur = cur.next
            cur.next = newnode

    #function to make the linked list object as iterable object
    def __iter__(self):
        cur = self.first
        while cur:
            yield cur.data
            cur = cur.next
# Linked List Iterators
ll = LinkedList()
ll.insert(2009)
ll.insert("welcome")
ll.insert("To")
ll.insert(179)
ll.insert("GPTA")
ll.insert(456.35)
ll.insert(545)
ll.insert(5)
print(“ The contents of linked list is:”)
for x in ll: # linked list object is traversed through for loop
    print(x, end= “  “)
```

OUTPUT:

The contents of linked list is:

2009 welcome To 179 GPTA 456.35 545 5

14. Implementation of Doubly linked list (DLL)(Traversing the Nodes, searching for a Node, Appending Nodes, Deleting Nodes):

```
class Node:
    def __init__(self, data = None):
        self.data = data
        self.next = None
        self.prev = None
class DoublyLinkedList:
    def __init__(self):
        self.first = None

    def insertatend(self, data):
      newnode = Node(data)
      if self.first==None:
          self.first =newnode
      else:
          cur=self.first
          while cur.next!=None:
              cur=cur.next
          cur.next=newnode
          newnode.prev=cur

    def removeFirst(self):
        if(self.first== None):
            print("list is empty")
        else:
            cur=self.first
            self.first=self.first.next
            print("the deleted item is",cur.data)

    def display(self):
        if(self.first== None):
            print("list is empty")
            return
        cur = self.first
        while(cur):
            print(cur.data, end = " ")
```

```
            cur = cur.next


    def search(self,item):
        if(self.first== None):
            print("list is empty")
            return
        cur = self.first
        found=False
        while cur != None and not found:
            if cur.data == item:
                found=True
            else:
                cur=cur.next
        if found:
            print("The data item is present in the list")
        else:
            print("The Data item is not present")

dll = DoublyLinkedList()
while(True):
    choice = int(input("\nEnter your choice 1-insert at end  2-delete from first 3-search 4-display 5.exit :"))
    if ( choice == 1):
        item = input("Enter the element to insert:")
        dll.insertatend(item)
        dll.display()
    elif ( choice == 2):
        dll.removeFirst()
        dll.display()
    elif ( choice == 3):
        item = input("Enter the element to search:")
        dll.search(item)
    elif ( choice == 4):
        dll.display()
    else:
        break
```

OUTPUT:

"C:\Users\LAXMI Y SHINGE\PycharmProjects\DSP\venv\Scripts\python.exe" "C:/Users/LAXMI Y SHINGE/PycharmProjects/DSP/doublylinklist.py"

Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :1
Enter the element to insert:karnataka
karnataka
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :1
Enter the element to insert:01
karnataka 01
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :1
Enter the element to insert:dept of cse
karnataka 01 dept of cse
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :1
Enter the element to insert:179
karnataka 01 dept of cse 179
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :4
karnataka 01 dept of cse 179
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :3
Enter the element to search:179
The data item is present in the list

Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :4
karnataka 01 dept of cse 179
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :02
the deleted item is karnataka
01 dept of cse 179
Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit :3
Enter the element to search:lms
The Data item is not present

Enter your choice 1-insert at end 2-delete from first 3-search 4-display 5-.exit : 5
Process finished with exit code 0

15 . Implementation of Circular linked list (CLL)(Traversing the Nodes, searching for a Node, Appending Nodes, and Deleting Nodes):

```
class Node:
    def __init__(self, data = None):
        self.data = data
        self.next = None
class CircularLinkedList:
    def __init__(self):
        self.first = None
    def insertatend(self,data):
        newnode=Node(data)
        if self.first==None:
            self.first=newnode
            self.first.next=newnode
        else:
            cur=self.first
            while cur.next!=self.first:
                cur=cur.next
            cur.next=newnode
            newnode.next=self.first

    def insertfromfirst(self,data):
        newnode=Node(data)
        if self.first==None:
            self.first=newnode
            self.first.next=newnode
        else:
            newnode.next=self.first
            cur=self.first
            while cur.next!=self.first:
                cur=cur.next
            cur.next=newnode
            self.first=newnode
```

```
def removefromend(self):
    if self.first==None:
        print("The list is empty")
    elif self.first.next==self.first:
        print("The deleted data is ",self.first.data)
        self.first=None
    else:
        cur=self.first
        prev=cur
        while cur.next!=self.first:
            prev=cur
            cur=cur.next
        prev.next=self.first
        print("The deleted data item is ",cur.data)

def display(self):
    if(self.first== None):
        print("list is empty")
        return
    cur = self.first
    while True:
        print(cur.data, end = " ")
        cur = cur.next
        if cur==self.first:
          return

def search(self,item):
    if(self.first== None):
        print("list is empty")
        return
    cur = self.first
    while cur.next != self.first:
        if cur.data == item:
            print("Item is present in the linked list")
            return
        else:
            cur = cur.next
    print("Item is not present in the linked list")
```

```
cll = CircularLinkedList()
while(True):
    choice = int(input("\nEnter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :"))
    if ( choice == 1):
        item = input("Enter the element to insert:")
        cll.insertatend(item)
        cll.display()
    elif ( choice == 2):
        cll.removefromend()
        cll.display()
    elif ( choice == 3):
        item = input("Enter the element to search:")
        cll.search(item)
    elif ( choice == 4):
        cll.display()
    elif ( choice == 5):
        item = input("Enter the element to insert:")
        cll.insertfromfirst(item)
        cll.display()
    else:
        break
```

OUTPUT:

```
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :1
Enter the element to insert:179
179
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :1
Enter the element to insert:gpta
179 gpta
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :5
Enter the element to insert:welcometo
welcometo 179 gpta
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :3
Enter the element to search:179
Item is present in the linked list

Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :3
Enter the element to search:cse
Item is not present in the linked list

Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :4
welcometo 179 gpta
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :2
The deleted data item is  gpta
welcometo 179
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :2
The deleted data item is  179
welcometo
Enter your choice 1-insertatend  2-deletefromend 3-search 4-display 5.Insertfromfirst 6.exit :6

Process finished with exit code 0
```

16. Implement stack data structure

```
# stack.py
class stack:
    def __init__ (self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def peek(self):
        return self.items[len(self.items) - 1]
    def size(self):
        return len(self.items)
    def display(self):
        return (self.items)

# stack_imp.py

import stack
s=stack.stack()
print(s.isEmpty())
print("push operations")
s.push(11)
s.push(12)
s.push(13)
print("size:",s.size())
print(s.display())
print("peek",s.peek())
print("pop operations")
print(s.pop())
print(s.pop())
print(s.display())
print("size:",s.size())
```

OUTPUT:

```
"C:\Users\LAXMI Y SHINGE\PycharmProjects\DSP\venv\Scripts\python.exe" "C:/Users/LAXMI Y SHINGE/PycharmProjects/DSP/stack_imp.py"
True
push operations
size: 3
[11, 12, 13]
peek 13
pop operations
13
12
[11]
size: 1

Process finished with exit code 0
```

17. Implement bracket matching using stack.

```
import stack
def check_brackets(statement):
        s = stack.stack()
        for token in statement:
            if token in "{[(":
                s.push(token)
            elif token in "}])":
                if s.isEmpty():
                    return False
                else:
                    left = s.pop()
                    if (token == "}" and left != "{") or \
                       (token == "]" and left != "[") or \
                       (token == ")" and left != "("):
                        return False
        return s.isEmpty()
# driver code to call the function check_brackets
stmt=input("Enter an expression:")
res=check_brackets(stmt)
if res==True:
   print(f"{stmt} is having balanced parantheses")
else:
  print(f"{stmt} is not having balanced parantheses")
```

OUTPUT:

Run1:

Enter an expression:(1*{2+3})

(1*{2+3}) is having balanced parantheses

Run2:

Enter an expression:(a+{b/[c//d}]))

(a+{b/[c//d}])) is not having balanced parantheses

18. Program to demonstrate recursive operations (factorial/ Fibonacci)

a) Factorial

```
def fact(n):
    if n == 0:
        return 1
    else:
      return (n * fact(n-1))
n=int(input("Enter the number:"))
print("The factorial of a number is:", fact(n))
```

b) Fibonacci Series

```
def fib(n):
    if (n == 0):
        return 0
    if n == 1 or n == 2:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)

n = int(input("Enter a number"))
print("Fibonacci series of %d numbers are :" % n, end=" ")
for i in range(0, n):
    print(fib(i), end=" ")
```

19. Implement solution for Towers of Hanoi.

```
def TowerOfHanoi(n , source, destination, auxiliary):
    if n==1:
        print ("Move disk 1 from source",source,"to destination",destination)
        return
    TowerOfHanoi(n-1, source, auxiliary, destination)
    print ("Move disk",n,"from source",source,"to destination",destination)
    TowerOfHanoi(n-1, auxiliary, destination, source)

n = int(input("Enter the number of disks:"))
TowerOfHanoi(n,'A','B','C')
```

OUTPUT:

```
Enter the number of disks:3

Move disk 1 from source A to destination B
Move disk 2 from source A to destination C
Move disk 1 from source B to destination C
Move disk 3 from source A to destination B
Move disk 1 from source C to destination A
Move disk 2 from source C to destination B
Move disk 1 from source A to destination B
```

20. Implement Queue Data Structure.

```
class Queue:
    def __init__(self):
        self.qlist=[]
    def IsEmpty(self):
        return len(self.qlist)==0
    def Enqueue(self,item):
        print("The data item inserted is",item)
        self.qlist.append(item)
    def Dequeue(self):
        if self.IsEmpty():
            print("Queue is Empty")
        else:
            del_item=self.qlist.pop(0)
            print("The deleted item is:",del_item)

    def display(self):
        if self.IsEmpty():
            print("Queue is empty")
        else:
            print(self.qlist)

queue=Queue()
while True:
    choice=int(input("Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit"))
    if choice==1:
        item=input("Enter the item to be inserted:")
        queue.Enqueue(item)
    elif choice==2:
        queue.Dequeue()
    elif choice==3:
        queue.display()
    else:
        break
```

OUTPUT:

```
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 2
Queue is Empty
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 3
Queue is empty
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 1
Enter the item to be inserted:179
The data item inserted is 179
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 1
Enter the item to be inserted:gpta
The data item inserted is gpta
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 1
Enter the item to be inserted:athani
The data item inserted is athani
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 3
['179', 'gpta', 'athani']
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 2
The deleted item is: 179
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 2
The deleted item is: gpta
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 2
The deleted item is: athani
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 2
Queue is Empty
Enter your choice: 1.Insert  2. Delele 3. Display 4. Exit 4
```

21. Implement Priority Queue Data Structure

```
class priorityQueueEntry:
    def __init__(self,value,p):
        self.value=value
        self.p=p
class PriorityQueue:
    def __init__(self):
        self.qlist=[]

    def IsEmpty(self):
        return len(self.qlist)==0

    def Queue_Length(self):
        return len(self.qlist)

    def Enqueue(self,value,priority):
        data_item=priorityQueueEntry(value,priority)
        self.qlist.append(data_item)

    def Dequeue(self):
        if self.IsEmpty():
            print("Cannot perform Dequeue operation")
            return
        else:
            highest_priority=self.qlist[0].p
            index=0
            for i in range(0,self.Queue_Length()):
                if highest_priority>self.qlist[i].p:
                    highest_priority=self.qlist[i].p
                    index=i
            del_item=self.qlist.pop(index)
            print("The deleted item is ",del_item.value)
```

```
    def display(self):
        if self.IsEmpty():
            print("Queue is empty")
        else:
            for x in range(0,self.Queue_Length()):
                print(self.qlist[x].value ,":" ,self.qlist[x].p)


pq = PriorityQueue()
while(True):
 print("1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit")
 choice = int(input("Enter your choice:"))
 if choice == 1:
    value = input("enter the item to insert:")
    priority = int(input("Enter the priority:"))
    pq.Enqueue(value,priority)
 elif choice == 2:
    pq.Dequeue()
 elif choice == 3:
    pq.display()
 elif choice == 4:
   print("length of queue is:",pq.Queue_Length())
 else:
    break
```

OUTPUT:

```
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:2
Cannot perform Dequeue operation
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:3
Queue is empty
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:1
enter the item to insert:White
Enter the priority:7
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:1
enter the item to insert:Blue
Enter the priority:0
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:4
length of queue is: 2
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:3
White : 7
Blue : 0
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:2
The deleted item is  Blue
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:2
The deleted item is  White
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:2
Cannot perform Dequeue operation
1:Enqueue 2:Dequeue 3:Display 4:Length 5:Exit
Enter your choice:5

Process finished with exit code 0
```

22. Implement Binary search tree and its operations using list.

```
class BSTNode:
    def __init__(self,value):
        self.data=value
        self.left=None
        self.right=None
class BinarySearchTree:
    def __init__(self):
        self.root=None

    def search(self, key):
        curNode = self.root
        while curNode is not None:
            if key == curNode.data:
                return True
            elif key < curNode.data:
                curNode = curNode.left
            else:
                curNode = curNode.right
        return False

    def delete(self, key):
        curNode = self.root
        parentNode = None
        while curNode is not None:
            if key == curNode.data:
                if temp == "Left":
                    parentNode.left = None
                else:
                    parentNode.right = None
                print(key, "Node Deleted")
                return True
            elif key < curNode.data:
                parentNode = curNode
                curNode = curNode.left
                temp = "Left"
```

```
                else:
                        parentNode = curNode
                        curNode = curNode.right
                        temp = "Right"
        print(key, "Node not found")
        return False

def insert(self, value):
        newNode = BSTNode(value)
        if self.root is None:
                self.root = newNode
        else:
                curNode = self.root
                while curNode is not None:
                        if value < curNode.data:
                                if curNode.left is None:
                                        curNode.left = newNode
                                        break
                                else:
                                        curNode = curNode.left
                        else:
                                if curNode.right is None:
                                        curNode.right = newNode
                                        break
                                else:
                                        curNode = curNode.right

def preorder(self, rt):
        print(rt.data, end="\t")
        if rt.left is not None:
                self.preorder(rt.left)
        if rt.right is not None:
                self.preorder(rt.right)
```

```
        def inorder(self, rt):
                if rt.left is not None:
                        self.inorder(rt.left)
                print(rt.data, end="\t")
                if rt.right is not None:
                        self.inorder(rt.right)

        def postorder(self, rt):
                if rt.left is not None:
                        self.postorder(rt.left)
                if rt.right is not None:
                        self.postorder(rt.right)
                print(rt.data, end="\t")

BT = BinarySearchTree()

ls = [25, 10, 35, 20, 65, 45, 24]
for i in ls:
   BT.insert(i)

print("\nPre-order")
BT.preorder(BT.root)
print("\nIn-order")
BT.inorder(BT.root)
print("\nPost-order")
BT.postorder(BT.root)
print("\n35 exists:", BT.search(35))
print("65 exists:", BT.search(65))
BT.delete(75)
BT.delete(24)
print("In-order")
BT.inorder(BT.root)
```

OUTPUT:-

```
Pre-order
25 10 20 24 35 65 45
In-order
10 20 24 25 35 45 65
Post-order
24 20 10 45 65 35 25
35 exists: True
65 exists: True
75 Node not found
24 Node Deleted
In-order
10 20 25 35 45
```

23. Implementation of DFS

```
class Stack:
    def __init__ (self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()

class BSTNode:
    def __init__(self, value):
        self.data = value
        self.left = None
        self.right = None
class BinarySearchTree:
    def __init__(self):
        self.root = None
    def insert(self, value):
            newNode = BSTNode(value)
            if self.root is None:
                self.root = newNode
            else:
                curNode = self.root
                while curNode is not None:
                    if value < curNode.data:
                        if curNode.left is None:
                            curNode.left = newNode
                            break
                        else:
                            curNode = curNode.left
                    else:
                        if curNode.right is None:
                            curNode.right = newNode
                            break
                        else:
                            curNode = curNode.right
```

```
def DFS(root):
    S =Stack()
    S.push(root)
    while S.isEmpty() != True:
        node=S.pop()
        print(node.data,end="\t")
        if node.right is not None:
            S.push(node.right)
        if node.left is not None:
            S.push(node.left)

BT = BinarySearchTree()

ls = [25,10,35,20,5,30,40]
for i in ls: BT.insert(i)
print("DFS Traversal")
DFS(BT.root)
```

OUTPUT:

```
DFS Traversal
25      10      5       20      35      30      40
```

24. Implementation of BFS

```
class Queue:
    def __init__(self):
        self.qlist=[]

    def IsEmpty(self):
        return len(self.qlist)==0

    def Enqueue(self,item):
        self.qlist.append(item)

    def Dequeue(self):
        if self.IsEmpty():
            print("Queue is Empty")
        else:
            return self.qlist.pop(0)

class BSTNode:
    def __init__(self, value):
        self.data = value
        self.left = None
        self.right = None


class BinarySearchTree:
    def __init__(self):
        self.root = None

    def insert(self, value):
        newNode = BSTNode(value)

        if self.root is None:
            self.root = newNode
        else:
            curNode = self.root
            while curNode is not None:
                if value < curNode.data:
                    if curNode.left is None:
                        curNode.left = newNode
                        break
                    else:
                        curNode = curNode.left
                else:
                    if curNode.right is None:
                        curNode.right = newNode
                        break
                    else:
                        curNode = curNode.right
```

```
def BFS(root):
   Q = Queue()
   Q.Enqueue(root)
   while Q.IsEmpty() != True:
      node=Q.Dequeue()
      print(node.data,end="\t")
      if node.left is not None:
         Q.Enqueue(node.left)
      if node.right is not None:
         Q.Enqueue(node.right)


BT = BinarySearchTree()
ls = [25,10,35,20,5,30,40]
for i in ls:
   BT.insert(i)

print("BFS Traversal")
BFS(BT.root)
```

OUTPUT:

```
BFS Traversal
25          10  35      5       20      30      40
```