MRCET CAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

Maisammaguda, Dhulapally, Komaplly, Secunderabad - 500100, Telangana State, India

LABORATORY MANUAL & RECORD

Name:.....

Roll No:.....Branch:.....

Year:.....Sem:.....

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION - UGC, GOVT. OF INDIA)

Affiliated to JNTUH; Approved by AICTE, NBA-Tier 1 & NAAC with A-GRADE | ISO 9001:2015

Maisammaguda, Dhulapally, Komaplly, Secunderabad - 500100, Telangana State, India

Certificate

Certified that this is the Bonafide Record of the Work Done by

Mr./Ms.....Roll.No.....of

B.Tech.....year..... Semester for Academic year.....

in.....Laboratory.

Date: Faculty Incharge HOD

Internal Examiner External Examiner

INDEX

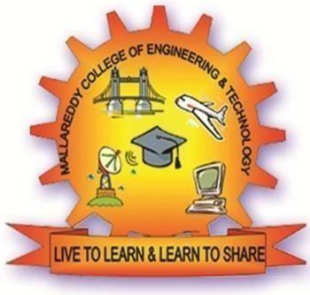
S.No	Date	Name of the Activity/Experiment	Grade/ Marks	Faculty Signature

INDEX

S.No	Date	Name of the Activity/Experiment	Grade/ Marks	Faculty Signature

DATA STRUCTURES LAB MANUAL

B.TECH
(R22A0583)



(II YEAR–I SEM)
(2024-25)

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

(CSE-AIML)

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12 (B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100, Telangana State, India

Department of Computational Intelligence

(CSE-AIML)

PROGRAMME EDUCATIONAL OBJECTIVES(PEOs)

PEO1:PROFESSIONALISM & CITIZENSHIP

To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.

PEO2:TECHNICAL ACCOMPLISHMENTS

To provide knowledge based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field.

PEO3:INVENTION, INNOVATION AND CREATIVITY

To make the students to design, experiment, analyze, interpret in the core field with the help of other multi-disciplinary concepts wherever applicable.

PEO4:PROFESSIONAL ETHICS

To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.

PEO5:HUMAN RESOURCE DEVELOPMENT

To graduate the students in building national capabilities in technology, education and research.

PROGRAMOUTCOMES (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design /development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety,and the cultural, societal, and environmental considerations.

4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the Impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of,and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team ,to manage projects and in multi disciplinary environments.

12. **Life long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

MALLAREDDYCOLLEGEOFENGINEERING&TECHNOLOGY

Maisammaguda,DhulapallyPost,ViaHakimpet,Secunderabad–500100

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

(CSE-AIML)

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details(Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.s
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

MALLA REDDY COLLEGE OF ENGINEERING AND TECHNOLOGY

B.TECH – II YEAR I –SEM **L/T/P/C**

-/-/3/1.5

(R22A0593)

DATA STRUCTURES USING PYTHON LAB

COURSE OBJECTIVES:

1. Understand basic data structures in python like Lists, Tuples, Dictionaries, Sets and Maps
2. Design and analyze simple linear data structures.
3. Identify and apply the suitable data structure for the given real world problem.
4. Design and analyze non linear data structures.
5. Gain knowledge in practical applications of data structures

List of Experiments

1. Write a Python program for class, Flower, that has three instance variables of type str, int, and float that respectively represent the name of the flower, its number of petals, and its price. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type, and retrieving the value of each type.
2. Develop an inheritance hierarchy based upon a Polygon class that has abstract methods area( ) and perimeter( ). Implement classes Triangle, Quadrilateral, Pentagon, that extend this base class, with the obvious meanings for the area( ) and perimeter( ) methods. Write a simple program that allows users to create polygons of the various types and input their geometric dimensions, and the program then outputs their area and perimeter.
3. Write a python program to implement Method Overloading and Method Overriding.
4. Write a program for Linear Search and Binary search
5. Write a program to implement Bubble Sort and Selection Sort
6. Write a program to implement Merge sort and Quick sort
7. Write a program to implement Stacks and Queues
8. Write a program to implement Singly Linked List
9. Write a program to implement Doubly Linked List
10. Write a python program to implement DFS & BFS graph Traversal Techniques.
11. Write a program to implement Binary Search Tree
12. Write a program to implement B+ Tree

DATA STRUCTURES LAB

Table of Contents

S.No	Name of the Experiment	Page No
1.	Write a Python program for class, Flower, that has three instance variables of type str, int,and float that respectively represent the name of the flower, its number of petals, and its price. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type, and retrieving the value of each type.	1
2.	Develop an inheritance hierarchy based upon a Polygon class that has abstract methods area( ) and perimeter( ). Implement classes Triangle, Quadrilateral, Pentagon, that extend this base class, with the obvious meanings for the area( ) and perimeter( ) methods. Write a simple program that allows users to create polygons of the various types and input their geometric dimensions, and the program then outputs their area and perimeter.	5
3.	Write a python program to implement Method Overloading and Method Overriding.	8
4.	Write a program for Linear Search and Binary search	11
5.	Write a program to implement Bubble Sort and Selection Sort	14
6	Write a program to implement Merge sort and Quick sort	16
7.	Write a program to implement Stacks and Queues	19
8.	Write a program to implement Singly Linked List	**25**
9.	Write a program to implement Doubly Linked List	33
10.	Write a python program to implement DFS & BFS graph Traversal Techniques.	41
11.	Write a program to implement Binary Search Tree	47
12.	Write a program to implement B+ Tree	53

1. Write a Python program for class, Flower, that has three instance variables of type str, int, and float, that respectively represent the name of the flower, its number of petals, and its price. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type, and retrieving the value of each type.

Program:

```
Class Flower:
#Common base class for all Flowers
    definit(self,petalName,petalNumber,petalPrice):self
        .name=petalName
        self.petals=petalNumberself.pri
        ce=petalPrice

    defsetName(self,petalName):self.name
        =petalName

    defsetPetals(self,petalNumber):self.
        petals=petalNumber

    defsetPrice(self,petalPrice):self.pr
        ice=petalPrice

    defgetName(self):retur
        nself.name

    defgetPetals(self):ret
        urnself.petals

    defgetPrice(self):retu
        rnself.price


#ThiswouldcreatefirstobjectofFlowerclassf1=Flo
wer("Sunflower",2,1000)
print("FlowerDetails:")print("
Name:",f1.getName())
print("Numberofpetals:",f1.getPetals())print("
Price:",f1.getPrice())

print("\n")

#ThiswouldcreatesecondobjectofFlowerclassf2=Flow
er("Rose",5,2000)
f2.setPrice(3333)f2.setPet
als(6)
print("FlowerDetails:")print("
Name:",f2.getName())
print("Numberofpetals:",f2.getPetals())print("
Price:",f2.getPrice())
```

Output:

Signature of the Faculty

Exercise Programs:

2. Develop an inheritance hierarchy based upon a Polygon class that has abstract methods area( ) and perimeter( ). Implement classes Triangle, Quadrilateral, Pentagon, that extend this base class, with the obvious meanings for the area() and perimeter() methods.Write a simple program that allows users to create polygons of the various types and input their geometric dimensions, and the program then outputs their area and perimeter.

Program:

```
From abc import abstract method,ABC
Meta import math

classPolygon(metaclass=ABCMeta):
    definit(self,side_lengths=[1,1,1],self._s   num_sides=  3):
        ide_lengths=side_lengthsself._num_siz
        es=3

    @abstractmethodde
    farea(self):
        pass

    @abstractmethod
    defperimeter(self):pass

    def___repr__(self):
        return(str(self._side_lengths))



classTriangle(Polygon):
    def_
        __init(self,side_lengths):super().
        init(side_lengths,3)self._perimete
        r=self.perimeter()self._area=self.


    defperimeter(self):return(sum(self.
        _side_lengths))

    defarea(self):

        #AreaofTriangle
        s=self._perimeter/2product=
        s
        foriinself._side_lengths:pro
            duct*=(s-i)
        returnproduct**0.5


classQuadrilateral(Polygon):
    def_
        __init(self,side_lengths):super().
        init(side_lengths,4)self._perimete
```

```
        self._area=self.area()


    defperimeter(self):return(sum(self._
        side_lengths))

    defarea(self):

        #AreaofanirregularQuadrilateralsemiperimeter=su
        m(self._side_lengths)/2
        returnmath.sqrt((semiperimeter-
                    self._side_lengths[0])*(semiperimeter-
                    self._side_lengths[1])*(semiperimeter-
                    self._side_lengths[2])*(semiperimeter-
                    self._side_lengths[3]))

classPentagon(Polygon):
    def_
        __init(self,side_lengths):super().
        init(side_lengths,5)self._perimete
        r=self.perimeter()self._area=self.
        area()

    defperimeter(self):return((self._sid
        e_lengths)*5)

    defarea(self):

        #AreaofaregularPentagona=se
        lf._side_lengths
        return(math.sqrt(5*(5+2*(math.sqrt(5))))*a*a)/4

#object  of
         Triangle1=Tr
iangle([1,2,2])print(t
1.perimeter(),t1.area
())

#objectofQuadrilateral
q1=Quadrilateral([1,1,1,1])p
rint(q1.perimeter(),q1.area()
```

Output:

Signature of the Faculty

Exercise Programs:

3. Write a python program to implement method overloading and method overriding.

Method Overloading

Method overloading is an OOPS concept which provides ability to have several methods having the same name with in the class where the methods differ in types or number of arguments passed.

Method overloading in Python

Method overloading in its traditional sense (as defined above) as exists in other languages like method overloading in Java doesn't exist in Python.

In Python if you try to overload a function by having two or more functions having the same name but different number of arguments only the last defined function is recognized, calling any other overloaded function results in an error.

Achieving method overloading

Since using the same method name again to overload the method is not possible in Python, so achieving method overloading in Python is done by having a single method with several parameters. Then you need to check the actual number of arguments passed to the method and perform the operation accordingly.

Program:

```
classOverloadDemo:
  #summethodwithdefaultasNoneforparametersdefsum(se
  lf,a=None,b=None,c=None):
    #Whenthreeparamsarepassed
    ifa!=Noneandb!=Noneandc!=None:s=a+
      b+c
      print('Sum=',s)
    #Whentwoparamsarepassedelif
    a!=Noneandb!=None:
      s=a+bprint('Sum='
      ,s)

od=OverloadDemo()od.sum(7,
8)
od.sum(7,8,9)
```

Output:

Method overriding-Polymorphism through inheritance

Method overriding provides ability to change the implementation of a method ina child class which is already defined in one of its super class. If there is a method in a super class and method having the same name and same number of arguments in a child class then the child class method is said to be overriding the parent class method.

When the method is called with parent class object, method of the parent class is executed. When method is called with child class object, method of the child class is executed. So the appropriate overridden method is called based on the object type, which is an example of Polymorphism.

Program:

```
classPerson:
  def
    ____init(self,name,age):sel
    f.name=name
    self.age=age

  defdisplayData(self):
    print('InparentclassdisplayDatamethod')print(self.n
    ame)
    print(self.age)

classEmployee(Person):
  def___init__(self,name,age,id):
    #callingconstructorofsuperclasssuper
    ()._______init
    __________(name,age)self.empId=id

  defdisplayData(self):
    print('InchildclassdisplayDatamethod')print(self.na
    me)
    print(self.age)print(s
    elf.empId)

#Personclassobject
person=Person('KarthikShaurya',26)person.displayData()
#Employeeclassobject
emp=Employee(''KarthikShaurya',26,'E317')emp.displayDat
a()
```

Output:

Signature of the Faculty

4. Write a program for Linear Search and Binary search

Linear Search Program:

```
deflinearSearch(target,List):positi
    on=0
    globaliterationsiterat
    ions=0
    whileposition<len(List):iterations+=
        1
        iftarget==List[position]:return
            position
        position+=1
    return-1

if___name___=='____main__':
    List=[1,2,3,4,5,6,7,8]
    target=3
    answer=linearSearch(target,List)ifan
    swer!=-1:
        print('Targetfoundatindex:',answer,'in',iterat
              ions,'iterations')
    else:
        print('Targetnotfoundinthelist')
```

Output:

BinarySearchProgram:

```
defbinarySearch(target,List):

    left=0
    right=len(List)-
    1globaliterationsiteratio
    ns=0

    whileleft<=right:iteratio
        ns+=1
        mid=(left+right)//2iftarg
        et==List[mid]:
            returnmid
        eliftarget<List[mid]:righ
            t=mid-1
        else:
            left=mid+1return-
    1

if___name___=='____main__':
    List=[1,2,3,4,5,6,7,8,9,11,12,13,              14]
    target=12
    answer=binarySearch(target,List)if(answer!=
    -1):
        print('Target',target,'foundatposition',it answer,'in',
              erations,'iterations')
    else:
        print('Targetnotfound')
```

Output:

Signature of the Faculty

Exercise Programs:

5. Write a program to implement Bubble Sort and Selection Sort

BubbleSortProgram:

```
defbubble_sort(alist):
    foriinrange(len(alist)-1,0,-
        1):no_swap=True
        forjinrange(0,i):
            ifalist[j+1]<alist[j]:
                alist[j],alist[j+1]=alist[j+1],alist[j]no_swap=False
        ifno_swap:
            return


alist=input('Enterthelistofnumbers:').split()alist=
[int(x)forxinalist]
bubble_sort(alist)print('Sorte
dlist:',alist)
```

Output:

SelectionSortProgram:

```
defselection_sort(alist):
    foriinrange(0,len(alist)-
        1):smallest=i
        forjinrange(i+1,len(alist)):ifalis
            t[j]<alist[smallest]:
                smallest=j
        alist[i],alist[smallest]=alist[smallest],alist[i]


alist=input('Enterthelistofnumbers:').split()alist=
[int(x)forxinalist]selection_sort(alist)
print('Sortedlist:',alist)
```

Output:

SignatureoftheFaculty

Exercise Programs:

6. Write a program to implement Mergesort and Quicksort

MergeSortProgram:

```
def merge_sort(alist,start,end):
    '''Sortsthelistfromindexesstarttoendifend    -1inclusive.'''
    -start>1:
        mid=(start+end)//2merge_sort(alist,st
        art,mid)merge_sort(alist,mid,end)merg
        e_list(alist,start,mid,end)


defmerge_list(alist,start,mid,end):left=a
    list[start:mid]
    right=alist[mid:end]k=
    start
    i=0
    j=0
    while(start+i<midandmid+j<end):if(left[i]<
        =right[j]):
            alist[k]=left[i]i=
            i+1
        else:
            alist[k]=right[j]
            j=j+1
        k=k+1
    ifstart+i<mid:whilek<e
        nd:
            alist[k]=left[i]i=
            i+1
            k=k+1
    else:
        whilek<end:
            alist[k]=right[j]
            j=j+1
            k=k+1


alist=input('Enterthelistofnumbers:').split()alist=[
int(x)forxinalist]
merge_sort(alist,0,len(alist))print('Sortedl
ist:',alist)
```

Output:

Signature of the Faculty

QuickSortProgram:

```
defquicksort(alist,start,end):
    '''Sortsthelistfromindexesstarttoend-1inclusive.'''ifend-
    start>1:
        p=partition(alist,start,end)qui
        cksort(alist,start,p)quicksort(
        alist,p+1,end)


defpartition(alist,start,end):pivot
    =alist[start]
    i=start+1
    j=end-1

    whileTrue:
        while(i<=jandalist[i]<=pivot):i=i+1
        while(i<=jandalist[j]>=pivot):j=j-1

        ifi<=j:
            alist[i],alist[j]=alist[j],alist[i]else
        :
            alist[start],alist[j]=alist[j],alist[start]returnj


alist=input('Enterthelistofnumbers:').split()alist=
[int(x)forxinalist]
quicksort(alist,0,len(alist))print('Sort
edlist:',alist)
```

Output:

Signature of the Faculty

7. Write a program to implement Stacks and Queues

StackProgram:

```
#CustomstackimplementationinPythonclassS
tack:

    #Constructortoinitializethestackdefi
    nit(self,size):
       self.arr=[None]*sizeself.ca
        pacity=sizeself.top=-1

    #Functiontoaddanelement`x`tothestackdefpush(sel
    f,x):
        ifself.isFull():
            print("StackOverflow!!Callingexit()…")exit(
            1)

        print("Inserting",x,"intothestack…")sel
        f.top=self.top+1self.arr[self.top]=x

    #Functiontopopatopelementfromthestackdefpop(sel
    f):
        #checkforstackunderflowifse
        lf.isEmpty():
            print("StackUnderflow!!Callingexit()…")exit
            (1)

        print("Removing",self.peek(),"fromthestack")

        #decreasestacksizeby1and(optionally)returnthepoppedelement
        top=self.arr[self.top]self.
        top=self.top-1returntop

    #Functiontoreturnthetopelementofthestackdefpeek(sel
    f):
        ifself.isEmpty():e
            xit(1)
        returnself.arr[self.top]

    #Functiontoreturnthesizeofthestackdefsize
    (self):
        returnself.top+1

    #Functiontocheckifthestackisemptyornotdefisempty(s
    elf):
        returnself.size()==0
```

```
#FunctiontocheckifthestackisfullornotdefisFull(
    self):
        returnself.size()==self.capacity


if___name__=='____main__':

    stack=Stack(3)

    stack.push(1)          #Inserting1inthestack#Inse
    stack.push(2)          rting2inthestack

    stack.pop()            #removingthetopelement(2)#remo
    stack.pop()            vingthetopelement(1)

    stack.push(3)          #Inserting3inthestack

    print("Topelementis",stack.peek())print(
    "Thestacksizeis",stack.size())

    stack.pop()            #removingthetopelement(3)

    #checkifthestackisemptyifstac
    k.isEmpty():
Output: print("Thestackisempty")else:
        print("Thestackisnotempty")
```

Queue Program:

```
#CustomqueueimplementationinPythonclassQueue:

    #Initializequeue
    def    init    (self,size):
        self.q=[None]*size
                                        #listtostorequeueelementsself.capacity
        =size
                                        #maximumcapacityofthequeueself.fron
        t=0#frontpointstothefrontelementinthequeueself.rear=-
        1#rearpointstothelastelementinthequeueself.count=0#currentsizeofthequeue


    #Functiontodequeuethefrontelementdefpop(self)
    :
        #checkforqueueunderflowifself.is
        Empty():
            print("QueueUnderflow!!Terminatingprocess.")exit(1)

        print("Removingelement…",self.q[self.front])

        self.front=(self.front+1)%self.capacityself.count=self.count-1


    #Functiontoaddanelementtothequeuedefappend(self,v
    alue):
        #checkforqueueoverflowifself.isF
        ull():
            print("Overflow!!Terminatingprocess.")exit(1)

        print("Insertingelement…",value)

        self.rear=(self.rear+1)%self.capacityself.q[self.rear]=value
        self.count=self.count+1


    #Functiontoreturnthefrontelementofthequeuedefpeek(self):
        ifself.isEmpty():
            print("QueueUnderFlow!!Terminatingprocess.")exit(1)

        returnself.q[self.front]


    #Functiontoreturnthesizeofthequeuedefsize(self):
        returnself.count
```

```
    #Functiontocheckifthequeueisemptyornotdefisempty(self):
        returnself.size()==0


    #Functiontocheckifthequeueisfullornotdefisfull(self):
        returnself.size()==self.capacity


if___name___=='____main__':

    #createaqueueofcapacity5q=Queue(5)

    q.append(1)
    q.append(2)
    q.append(3)

    print("Thequeuesizeis",q.size())print(
    "Thefrontelementis",q.peek())q.pop()
    print("Thefrontelementis",q.peek())

    q.pop()
    q.pop()

    ifq.isEmpty():
        print("Thequeueisempty")else:
        print("Thequeueisnotempty")
```

Output:

Signature of the Faculty

8. Write a program to implement Singly Linked List

Program:

```
importos
fromtypingimportNewType


class_Node:'''
        CreatesaNodewithtwofields:
        1. element(accesedusing._element)
        2. link(accesedusing._link)'''
        __slots____='_element','_link'

        def____init(self,element,link):'''
                Initialses_elementand_linkwithelementandlinkrespectively.
                '''
                self._element=elementself._link=li
                nk


classLinkedList:'''
        Consistsofmemberfuntionstoperformdifferentoperationsonthelinkedli
        st.
        '''

        def____init____(self):'''
                Initialseshead,tailandsizewithNone,Noneand0respectively.
                '''
                self._head=Noneself._
                tail=Noneself._size=0

        def____len(self):'''
                Returnslengthoflinkedlist'''
                returnself._size

        defisempty(self):'''
                ReturnsTrueiflinkedlistisempty,otherwiseFalse.'''
                returnself._size==0

        defaddLast(self,e):'''
```

```
        Addsthepassedelementattheendofthelinkedlist.'''
        newest=_Node(e,None)

        ifself.isempty():self._head=newest
        else:
            self._tail._link=newest

        self._tail=newestself._size+=1

    defaddFirst(self,e):'''
        Addsthepassedelementatthebeginningofthelinkedlist.
        '''
        newest=_Node(e,None)

        ifself.isempty():self._head=newest
            self._tail=newest
        else:
            newest._link=self._headself._head=ne
            west
        self._size+=1

    defaddAnywhere(self,e,index):'''
        Addsthepassedelementatthepassedindexpositionofthelinkedlist.
        '''
        newest=_Node(e,None)

        i=index-
        1p=self._head

        ifself.isempty():self.addFirst(e)
        else:
            foriinrange(i):p=p._lin
                k
            newest._link=p._linkp._link
            =newest
            print(f"AddedItematindex{index}!\n\n")self._size+=1


    defremoveFirst(self):'''
        Removeselementfromthebeginningofthelinkedlist.Returnstheremovedeleme
        nt.
        '''
        ifself.isempty():
            print("ListisEmpty.Cannotperformdeletionoperation.")
            return
```

```
        e=self._head._elementself._head=self.
        _head._linkself._size=self._size-1

        ifself.isempty():self._tail=None

        returne

    defremoveLast(self):'''
        Removeselementfromtheendofthelinkedlist.Returnstheremovedel
        ement.
        '''
        ifself.isempty():
            print("ListisEmpty.Cannotperformdeletionoperation.")
            return

        p=self._head
        ifp._link==None:e=p._e
            lement
            self._head=Noneelse:
            whilep._link._link!=None:p=p._lin
                k
            e=p._link._elementp._link=
            Noneself._tail=p

        self._size=self._size-1returne

    defremoveAnywhere(self,index):'''
        Removeselementfromthepassedindexpositionofthelinkedlist.Returnsthere
        movedelement.
        '''
        p=self._headi=ind
        ex-1

        ifindex==0:
            returnself.removeFirst()elifinde
        x==self._size-1:
            returnself.removeLast()else:
            forxinrange(i):p=p._li
                nk
            e=p._link._elementp._link=p._li
            nk._link

        self._size-
        =1returne
```

```
    defdisplay(self):

        Utilityfunctiontodisplaythelinkedlist.'''
        ifself.isempty()==0:p=self._he
            adwhilep:
                print(p._element,end='--
                >')p=p._link
            print("NULL")els
        e:
            print("Empty")

    defsearch(self,key):'''
        Searchesforthepassedelementinthelinkedlist.Returnstheindexpositioniff
        ound,else-1.
        '''
        p=self._headinde
        x=0whilep:
            ifp._element==key:returnin
                dex
            p=p._linkinde
            x+=1
        return -1

defoptions():'''
    PrintsMenuforoperations'''
    options_list=['AddLast','AddFirst','AddAnywhere',
                  'RemoveFirst','RemoveLast','RemoveAnywhere','DisplayList'
                  ,'PrintSize','Search','Exit']

    print("MENU")
    fori,optioninenumerate(options_list):print(f'{i+1}.{opt
        ion}')

    choice=int(input("Enterchoice:"))returnchoice
defswitch_case(choice):'''
    SwitchCaseforoperations'''
    ifchoice==1:
        elem=int(input("EnterItem:"))L.addLast(e
        lem)
        print("AddedItematLast!\n\n")

    elifchoice==2:
        elem=int(input("EnterItem:"))L.addFirst(e
        lem)
        print("AddedItematFirst!\n\n")
```

```
    elifchoice==3:
        elem=int(input("EnterItem:"))index
        =int(input("EnterIndex:"))L.addAny
        where(elem,index)

    elifchoice==4:
        print("RemovedElementfromFirst:",L.removeFirst())

    elifchoice==5:
        print("RemovedElementfromlast:",L.removeLast())

    elifchoice==6:
        index=int(input("EnterIndex:"))
        print(f"RemovedItem:{L.removeAnywhere(index)}!\n\n")

    elifchoice==7:print("List:"
        ,end='')L.display()
        print("\n")

    elifchoice==8:print("Size:"
        ,len(L))print("\n")

    elifchoice==9:
        key=int(input("Enteritemtosearch:"))ifL.se
        arch(key)>=0:
            print(f"Item{key}foundatindexposition
{L.search(key)}\n\n")el
        se:
            print("Itemnotinthelist\n\n")

    elifchoice==10:imports
        yssys.exit()

######################################################################

if___name__=='___main
    _':L=LinkedList()
    whileTrue:
        choice=options()switch_case(choi
        ce)
```

Output:

Signature of the Faculty

Exercise Programs:

9. Write a program to implement Doubly Linked List

Program:

```
importos

class_Node:'''
    CreatesaNodewiththreefields:
    1. element(accessedusing._element)
    2. link(accessedusing._link)
    3. prev(accessedusing._prev)'''
    __slots__='_element','_link','_prev'

    definit___(self,element,link,prev):'''
        Initialses_element,_linkand_prevwithelement,linkandprevrespectively.
        '''
       -self._element=elementse
        lf._link=linkself._prev
        =prev

classDoublyLL:'''
    Consistsofmemberfuntionstoperformdifferentoperation
    sonthedoublylinkedlist.
    '''

    definit
        _____(self):
        '''
        Initialiseshead,tailandsizewithNone,Noneand0respectively.
        '''
       -self._head=Nonese
        lf._tail=Noneself
        ._size=0

    deflen
        _____(self):
        '''
        Returnslengthoflinkedlist'''
       -returnself._size

    defisempty(self):'
        ''
        ReturnsTrueifdoublylinkedlistisempty,otherwiseFalse.
        '''
        returnself._size==0

    defaddLast(self,e):'''
        Addsthepassedelementattheendofthedoublylinkedlist.
        '''
```

```
        newest=_Node(e,None,None)ifself.ise

        mpty():
            self._head=newestelse:
            self._tail._link=newestnewest._prev=s
            elf._tail
        self._tail=newestself._size+=1

    defaddFirst(self,e):'''
        Addsthepassedelementatthebeginningofthedoublylinkedlist.
        '''
        newest=_Node(e,None,None)

        ifself.isempty():self._head=newest
            self._tail=newest
        else:
            newest._link=self._headself._head._pr
            ev=newest
        self._head=newestself._size+=1

    defaddAnywhere(self,e,index):'''
        Addsthepassedelementatthepassedindexpositionofthedoublylinkedlist.
        '''
        ifindex>=self._size:
            print(f'Indexvalueoutofrange,itshouldbebetween0-{self._size-1}')
        elifself.isempty():
            print("Listwasempty,itemwillbeaddedattheend")self.addLast(e)
        elifindex==0:self.addFirst(e)
        elifindex==self._size-1:self.addLast(e)
        else:
            newest=_Node(e,None,None)p=self._
            head
            for_inrange(index-1):p=p._link
            newest._link=p._linkp._link.
            _prev=newestnewest._prev=
            pp._link=newestself._size+=
            1

    defremoveFirst(self):'''
        Removeselementfromthebeginningofthedoublylinkedlist.Returnstheremove
        delement.
        '''
```

```
        ifself.isempty():
            print('Listisalreadyempty')return
        e=self._head._elementself._head=self.
        _head._linkself._size-=1

        ifself.isempty():self._tail=None
        else:
            self._head._prev=Nonereturne

    defremoveLast(self):'''
        Removeselementfromtheendofthedoublylinkedlist.Returnstheremovedeleme
        nt.
        '''
        ifself.isempty():
            print("Listisalreadyempty")return
        e=self._tail._elementself._tail=self._ta
        il._prevself._size-=1

        ifself.isempty():self._head=None
        else:
            self._tail._link=Nonereturne

    defremoveAnywhere(self,index):'''
        Removeselementfromthepassedindexpositionofthedoublylinkedlist.
        Returnstheremovedelement.'''
        ifindex>=self._size:
            print(f'Indexvalueoutofrange,itshouldbebetween0-{self._size-1}')
        elifself.isempty():print("Listisem
            pty")
        elifindex==0:
            returnself.removeFirst()elifinde
        x==self._size-1:
            returnself.removeLast()else:
            p=self._head
            for_inrange(index-1):p=p._link
            e=p._link._elementp._link=p._li
            nk._linkp._link._prev=pself._siz
            e-=1
        returne
```

```
    defdisplay(self):'''
        Utilityfunctiontodisplaythedoublylinkedlist.'''
        ifself.isempty():print("ListisEmpt
            y")return

        p=self._headprint("NULL<--
        >",end='')whilep:
            print(p._element,end="<--
            >")p=p._link
        print("NULL")

        print(f"\nHead:{self._head._element},Tail:
            {self._tail._element}")

defoptions():'''
    PrintsMenuforoperations'''
    options_list=['AddLast','AddFirst','AddAnywhere',
                  'RemoveFirst','RemoveLast','RemoveAnywhere','DisplayList'
                  ,'Exit']

    print("MENU")
    fori,optioninenumerate(options_list):print(f'{i+1}.{opt
        ion}')

    choice=int(input("Enterchoice:"))returnchoice


defswitch_case(choice):'''
    SwitchCaseforoperations'''
    os.system('cls')ifchoic
    e==1:
        elem=int(input("EnterItem:"))DL.addLast
        (elem)
        print("AddedItematLast!\n\n")

    elifchoice==2:
        elem=int(input("EnterItem:"))DL.addFirst
        (elem)
        print("AddedItematFirst!\n\n")

    elifchoice==3:
        elem=int(input("EnterItem:"))index=int(input
        ("EnterIndex:"))DL.addAnywhere(elem,index
        )
```

```
    elifchoice==4:
        print("RemovedElementfromFirst:",DL.removeFirst())

    elifchoice==5:
        print("RemovedElementfromlast:",DL.removeLast())

    elifchoice==6:
        index=int(input("EnterIndex:"))
        print(f"RemovedItem:{DL.removeAnywhere(index)}!\n\n")

    elifchoice==7:print("L
        ist:")DL.display()
        print("\n")

    elifchoice==8:impo
        rtsyssys.exit(
        )

#####################################################################

if__name___=='___main
    ':DL=DoublyLL()
    whileTrue:
        choice=options()switch_case(choi
        ce)
```

Output:

Signature of the Faculty

Exercise Programs:

10. Write a python program to implement DFS & BFS graph traversal

Techniques BREADTH FIRST SEARCH TRAVERSAL
#Python3ProgramtoprintBFStraversal#fromagivensou
rcevertex.BFS(ints)#traversesverticesreachablefroms.fr
omcollectionsimportdefaultdict

#Thisclassrepresentsadirectedgraph#usingadjacencyli
strepresentationclassGraph:

```
    #Constructor
    def __init__(self):

        #defaultdictionarytostoregraphself.graph=default
        dict(list)

    #functiontoaddanedgetograph
#Makealistvisited[]tocheckifanodeisalreadyvisitedornotdefaddEdge(self,u,v):
        self.graph[u].append(v)self.visited=[]

    #FunctiontoprintaBFSofgraphdefBFS(self,s)
    :

        #CreateaqueueforBFSqueue=[]

        #Addthesourcenodein#visiteda
        ndenqueueitqueue.append(s)self
        .visited.append(s)

        whilequeue:

            #Dequeueavertexfrom#queuea
            ndprintit
            s=queue.pop(0)print(s,en
            d="")

            #Getalladjacentverticesofthe#dequeuedvertexs.I
            faadjacent#hasnotbeenvisited,thenaddit#invisite
            dandenqueueit
            foriinself.graph[s]:ifinotinvisited:
                    queue.append(i)self.visited.append(s)
```

```
#Drivercode

#Createagraphgivenin#theabove
diagram
g=Graph()g.addEdg
e(0,1)
g.addEdge(0,2)
g.addEdge(1,2)
g.addEdge(2,0)
g.addEdge(2,3)
g.addEdge(3,3)

print("FollowingisBreadthFirstTraversal"+
"(startingfromvertex2)")
g.BFS(2)
```

Output:

Signature of the Faculty

Program to implement DFS & BFS graph traversal

```
#fromagivengraph
fromcollectionsimportdefaultdict


#Thisclassrepresentsadirectedgraphusing#adjacencylistrep
resentation
classGraph:

        #Constructor
        def __init__(self):

                #Defaultdictionarytostoregraphself.graph=def
                aultdict(list)


        #FunctiontoaddanedgetographdefaddEdg
        e(self,u,v):
                self.graph[u].append(v)


        #AfunctionusedbyDFS
        defDFSUtil(self,v,visited):

                #Markthecurrentnodeasvisited#andprintit
                visited.add(v)print(v,end
                ='')

                #Recurforallthevertices#adjacentto
                thisvertex
                forneighbourinself.graph[v]:ifneighbourn
                        otinvisited:
                                self.DFSUtil(neighbour,visited)


        #ThefunctiontodoDFStraversal.Ituses#recursiveDFSUti
        l()
        defDFS(self,v):

                #Createasettostorevisitedverticesvisited=set()

                #Calltherecursivehelperfunction#toprintDFSt
                raversalself.DFSUtil(v,visited)
```

```
#Driver'scode
if  —name—  ==" —main—
            ":g=Graph()

    print("FollowingisDepthFirstTraversal(startingfromvertex


    #Functioncallg.DFS(2)


2)")
g.addEdge(0,  1)
g.addEdge(0,  2)
g.addEdge(1,  2)
g.addEdge(2,  0)
g.addEdge(2,  3)
g.addEdge(3,  3)
```

Output:

Signature of the Faculty

11. Write a program to implement Binary Search Tree

Program:

```
###BinarySearchTreeclassbin
arySearchTree:
        definit(self,val=None):self.val
                                =
                                valself.left=N
                oneself.right=None

        definsert(self,val):
                #checkifthereisnorootif(self.val==
                None):
                        self.val=val
                #checkwheretoinsertelse:
                        #checkforduplicatethenstopandreturn
ifval==self.val:return'noduplicatesallowedinbinarysearchtree'
                        #checkifvaluetobeinserted<currentNode'svalueif(val<self.val):
    #checkifthereisaleftnodetocurrentNodeiftruethenrecurse
                                if(self.left):self.left.insert(val)
                                #insertwhereleftofcurrentNodewhencurrentNode.left=None
                                else:self.left=binarySearchTree(val)

                        #samestepsasaboveheretheconditionwecheckisvaluetobe#inserted>c
                         urrentNode'svalue
                        else:
                                if(self.right):self.right.insert(val)
                                else:self.right=binarySearchTree(val)


        defbreadthFirstSearch(self):current
                Node=selfbfs_list=[]
                queue=[]queue.insert(0,currentNode)while(
                len(queue)>0):
                        currentNode=queue.pop()bfs_list.append(currentNode.val)if
                        (currentNode.left):
                                queue.insert(0,currentNode.left)if(currentNode.right):
                                queue.insert(0,currentNode.right)returnbfs_lis

                t

        #Inordermeansfirstleftchild,thenparent,atlastrightchild
        defdepthFirstSearch_INorder(self):returnself.tr
                averseInOrder([])
```

```
#Preordermeansfirstparent,thenleftchild,atlastrightchild
defdepthFirstSearch_PREorder(self):returnself.
    traversePreOrder([])

#Postordermeansfirstleftchild,thenrightchild,atlastparent
defdepthFirstSearch_POSTorder(self):returnsel
    f.traversePostOrder([])

deftraverseInOrder(self,lst):if(self.left):
        self.left.traverseInOrder(lst)lst.append(self.val)
    if(self.right):self.right.traverseInOrder(lst)
    returnlst

deftraversePreOrder(self,lst):lst.append(self.val)
    if(self.left):self.left.traversePreOrder(lst)
    if(self.right):self.right.traversePreOrder(lst)
    returnlst

deftraversePostOrder(self,lst):if(self.left):
        self.left.traversePostOrder(lst)if(self.right)
    :
        self.right.traversePostOrder(lst)lst.append(self.val)
    returnlst

deffindNodeAndItsParent(self,val,parent=None):
    #returningthenodeanditsparentsowecandeletethe#nodeandreconstructthe
    treefromitsparent
    ifval==self.val:returnself,parentif(val<self.val):
        if(self.left):
            returnself.left.findNodeAndItsParent(val,self)else:return'Notfou
        nd'
    else:
        if(self.right):
            returnself.right.findNodeAndItsParent(val,self)else:return'Notfo
        und'

#deleteinganodemeanswehavetorearrangesomepartofthetree
defdelete(self,val):
    #checkifthevaluewewanttodeleteisinthetreeif(self.findNodeAndItsParent(val)=
    ='Notfound'):return'Nodeisnotintree'
    #wegetthenodewewanttodeleteanditsparent-
    node#fromfindNodeAndItsParentmethod
    deleteing_node,parent_node=self.findNodeAndItsParent(val)#checkho
    wmanychildrennodesdoesthenodewearegoing#todeletehavebytraverse
    PreOrderfromthedeleteing_nodenodes_effected=deleteing_node.traver
    sePreOrder([])
```

```
#iflen(nodes_effected)==1means,thenodetobedeleteddoesn't#haveanychildren
#sowecanjustcheckfromitsparentnodetheposition(leftor#right)ofnodewewan
ttodelete
#andpointthepositionto'None'i.enodeisdeleted

if(len(nodes_effected)==1):
        if(parent_node.left.val==deleteing_node.val):parent_node.left=None
        else:parent_node.right=Nonereturn'S
        uccesfullydeleted'
#iflen(nodes_effected)>1whichmeansthenodeweare#goingtodelete
has'children',
#sothetreemustberearrangedfromthedeleteing_nodeelse:
        #ifthenodewewanttodeletedoesn'thaveanyparent#meansthenodeto
        bedeletedis'root'nodeif(parent_node==None):
                nodes_effected.remove(deleteing_node.val)#makethe'
                root'nodeei.eself
                 value,left,righttoNone,#thismeansweneedtoimplementane
                 wtreeagainwithout#thedeletednode
                self.left=Noneself.right
                =Noneself.val=None

                 #constructionofnewtreefornodein
                nodes_effected:
                        self.insert(node)
                return'Succesfullydeleted'

        #ifthenodewewanttodeletehasaparent#traversefromparen
        t_node
        nodes_effected=parent_node.traversePreOrder([])

        #deletingthenode
        if(parent_node.left==deleteing_node):parent_node.left
        =None
        else:parent_node.right=None

        #removeingtheparent_node,deleteing_nodeandinserting#thenodes_effe
        ctedinthetreenodes_effected.remove(deleteing_node.val)nodes_effected.
        remove(parent_node.val)
        fornodeinnodes_effected:self.insert(no
                de)
```

return'Successfullydeleted'

**bst=binarySearchTree()bst.inse
rt(7)bst.insert(4)bst.insert(9)bst.i
nsert(0)bst.insert(5)bst.insert(8)
bst.insert(13)**

```
#           7
#         /   \
#       /       \
#      4          9
#     /\         /  \
#    0   5      8    13

#INorder-
usefulinsortingthetreeinascendingorderprint('INorder:',bst
.depthFirstSearch_INorder())

#PREorder-
usefulinreconstructingatreeprint('PREorder:',bst.depth
FirstSearch_PREorder())

#POSTorder-
usefulinfindingtheleafnodesprint('POSTorder:',bst.dept
hFirstSearch_POSTorder())

print(bst.delete(5))
print(bst.delete(9))
print(bst.delete(7))

#afterdeleting
```

Output:

Signature of the Faculty

12 Write a program for implementing B+ Tree

```
Import math
#NodecreationclassNo
de:
    def init__ __
        (self,order):self.order=orde
        rself.values=[]self.keys=[]self.n
        extKey=Noneself.parent=None
        self.check_leaf=False

    #Insertattheleaf
    definsert_at_leaf(self,leaf,value,key):if(self.values):
        temp1=self.values
        foriinrange(len(temp1)):if(value==te
            mp1[i]):
                self.keys[i].append(key)break
            elif(value<temp1[i]):
                self.values=self.values[:i]+[value]+
    self.values[i:]self.ke
                self.keys=self.keys[:i]+[[key]]+break
    ys[i:]

            elif(i+1==len(temp1)):self.values.append(val
                ue)self.keys.append([key])break

        self.values=[value]self.keys=[[key]]
else:

    #BplustreeclassBplus
    Tree:
        def init
            (self,order):self.root=Node(orde
        r)self.root.check_leaf=True

        __ __
        #Insertoperation
        definsert(self,value,key):value=str(value)
        old_node=self.search(value)old_node.insert_at_leaf(old_node,value,key)

        if(len(old_node.values)==old_node.order):node1=Node(old_node
            .order)
```

```
            node1.check_leaf=Truenode1.parent=old_
            node.parent
            mid=int(math.ceil(old_node.order/2))-
            1node1.values=old_node.values[mid+1:]node1.keys=old_node
            .keys[mid+1:]node1.nextKey=old_node.nextKeyold_node.valu
            es=old_node.values[:mid+1]old_node.keys=old_node.keys[:mi
            d+1]old_node.nextKey=node1
            self.insert_in_parent(old_node,node1.values[0],node1)

#Searchoperationfordifferentoperationsdefsearch(self,value):
    current_node=self.rootwhile(current_node.check_leaf==
    False):
            temp2=current_node.valuesforiinran
            ge(len(temp2)):
                    if(value==temp2[i]):
                            current_node=current_node.keys[i+1]break
                    elif(value<temp2[i]):
                            current_node=current_node.keys[i]break
                    elif(i+1==len(current_node.values)):current_node=current_node.
                            keys[i+1]break
    returncurrent_node

#Findthenode
deffind(self,value,key):l=self.search(
    value)
    fori,iteminenumerate(l.values):ifitem==value:
                    ifkeyinl.keys[i]:returnTrue
                    else:re
                            returnFalse
    turnFalse

#Insertingattheparent
definsert_in_parent(self,n,value,ndash):if(self.root==n):
            rootNode=Node(n.order)rootNode.
            values=[value]rootNode.keys=[n,nd
            ash]self.root=rootNoden.parent=roo
            tNodendash.parent=rootNodereturn
```

```
            parentNode=n.parenttemp3=paren
            tNode.keys
            foriinrange(len(temp3)):if(temp3[i]
                  ==n):
                        parentNode.values=parentNode.values[:i]+\[value]+parentNode.v
                              alues[i:]
                        parentNode.keys=parentNode.keys[:i+
                                                                  1]+[ndash]
      +parentNode.keys[i+1:]
                        if(len(parentNode.keys)>parentNode.order):parentdash=Node(parent
                              Node.order)parentdash.parent=parentNode.parent
                              mid=int(math.ceil(parentNode.order/2))-
                              1parentdash.values=parentNode.values[mid+1:]parentdash.keys=
                              parentNode.keys[mid+1:]value_=parentNode.values[mid]
                              if(mid==0):
                                    parentNode.values=parentNode.values[:mid+
      1]
 else:
                              parentNode.values=parentNode.values[:mid]
                              parentNode.keys=parentNode.keys[:mid+1]forjinparentNode
                              .keys:
                                    j.parent=parentNodeforjinparen
                              tdash.keys:
                                    j.parent=parentdashself.insert_in_parent(parentNode,
                              value_,
      parentdash)#Pri

   ntthetree
   defprintTree(tree):lst=[tree.r
      oot]level=[0]
      leaf=Noneflag=
      0
      lev_leaf=0

      node1=Node(str(level[0])+str(tree.root.values))while(len(lst)!=0):
          x=lst.pop(0)
          lev=level.pop(0)
          if(x.check_leaf==False):
                fori,iteminenumerate(x.keys):print(item.values)
else:
                fori,iteminenumerate(x.keys):print(item.values)

          if(flag==0):

     lev_leaf=lev
    leaf=x
```

```
                    flag=1


record_len=3
bplustree=BplusTree(record_len)bplustree.insert('5','33')
bplustree.insert('15','21')
bplustree.insert('25','31')
bplustree.insert('35','41')
bplustree.insert('45','10')printTree(bplustree)

if(bplustree.find('5','34')):print("Found")
else:
    print("Notfound")
```

Output:

Signature of the Faculty