Diploma Engineering

Laboratory Manual

(Data Structures and Applications)
(1333203)

[Information and Communication Technology, Semester III]

Enrolment No	
Name	
Branch	
Academic Term	
Institute	

Directorate of Technical Education
Gandhinagar - Gujarat

DTE's Vision:

- To provide globally competitive technical education;
- Remove geographical imbalances and inconsistencies;
- Develop student friendly resources with a special focus on girls' education and support to weaker sections;
- Develop programs relevant to industry and create a vibrant pool of technical professionals.

Institute's Vision:

Institute's Mission:

Department's Vision:

Department's Mission:

Name of institute

Certificate

This is to certify that Mr. /Ms …….. Enrolment No. …….. of 3rd Semester of *Diploma in Information and Communication Technology* of the Institute …….. (GTU Code) has satisfactorily completed the term work in course …….. for the academic year: …… Term: Odd/Even *prescribed* in the GTU curriculum.

Place:

Date: ……..

Signature of Course Faculty **Head of the Department**

Preface

The primary aim of any laboratory/Practical/field work is enhancement of required skills as well as creative ability amongst students to solve real time problems by developing relevant competencies in psychomotor domain. Keeping in view, GTU has designed competency focused outcome-based curriculum -2021 (COGC-2021) for Diploma engineering programmes. In this more time is allotted to practical work than theory. It shows importance of enhancement of skills amongst students and it pays attention to utilize every second of time allotted for practical amongst Students, Instructors and Lecturers to achieve relevant outcomes by performing rather than writing practice in study type. It is essential for effective implementation of competency focused outcome- based Green curriculum-2021. Every practical has been keenly designed to serve as a tool to develop & enhance relevant industry needed competency in each and every student. These psychomotor skills are very difficult to develop through traditional chalk and board content delivery method in the classroom. Accordingly, this lab manual has been designed to focus on the industry defined relevant outcomes, rather than old practice of conducting practical to prove concept and theory.

By using this lab manual, students can read procedure one day in advance to actual performance day of practical experiment which generates interest and also, they can have idea of judgement of magnitude prior to performance. This in turn enhances predetermined outcomes amongst students. Each and every Experiment /Practical in this manual begins by competency, industry relevant skills, course outcomes as well as practical outcomes which serve as a key role for doing the practical. The students will also have a clear idea of safety and necessary precautions to be taken while performing experiment.

This manual also provides guidelines to lecturers to facilitate student-centred lab activities for each practical/experiment by arranging and managing necessary resources in order that the students follow the procedures with required safety and necessary precautions to achieve outcomes. It also gives an idea that how students will be assessed by providing Rubrics.

Data structures are fundamental components and play a crucial role in managing and organizing data efficiently. Python, a popular programming language, provides powerful tools and libraries for implementing various data structures. Understanding data structures is essential for solving complex problems and optimizing algorithm performance. This course on Data Structures using Python aims to equip students with the necessary knowledge and skills to work with different data structures such as arrays, linked lists, stacks, queues, trees, and graphs. Through hands-on exercises and practical examples, students will learn how to implement, manipulate, and analyse data structures

using Python programming. This course will serve as a solid foundation for students interested in pursuing advanced topics like algorithm design, data analysis, and artificial intelligence in future semesters. By the end of this course, students will have a comprehensive understanding of data structures and the ability to leverage Python's capabilities to create efficient and scalable solutions for real-world problems.

Although we try our level best to design this lab manual, but always there are chances of improvement. We welcome any suggestions for improvement.

Programme Outcomes (POs):

1. **Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, science and engineering fundamentals and engineering specialization to solve the *engineering* problems.

2. **Problem analysis**: Identify and analyse well-defined *engineering* problems using codified standard methods.

3. **Design/ development of solutions:** Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

4. **Engineering Tools, Experimentation and Testing:** Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

5. **Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.

6. **Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.

7. **Life-long learning:** Ability to analyse individual needs and engage in updating in the context of technological changes *in field of engineering.*

Practical Outcome - Course Outcome Matrix

Course Outcomes (COs):

CO1: Understand the basic concepts of data structures, analysis terms, OOPs concepts, And comprehend the concept of recursive functions.

CO2: Apply basic operations on stack, queue data structures.

CO3: Apply basic operations on linked list data structure.

CO4: Apply different sorting and searching algorithms to the small data sets.

CO5: Illustrate algorithms to insert, delete and searching a node in tree.

S. No.	Practical Outcome/Title of experiment	CO1	CO2	CO3	CO4	CO5
1.	Design a student class for reading and displaying the student information, the getInfo () and displayInfo () methods will be used respectively. Where getInfo() will be a private method.	√	-	-	-	-
2.	Design a class Complex for adding the two complex numbers and show the use of constructor.	√	-	-	-	-
3.	Write Python class to represent a bank account. The class contains account holder name, account number, type of account and balance amount as the data members. The member functions are as follows: to initialize the data members, to deposit an amount, to withdraw an amount after checking balance, and to display name and balance.	√	-	-	-	-
4.	Write Python class to represent a Cricketer. The class contains name of cricketer, team name and run as the data members. The member functions are as follows: to initialize the data members, to set run and display run.	√	-	-	-	-
5.	Implement push and pop algorithms of stack using list.	-	√	-	-	-
6.	Implement a program to convert infix notation to postfix notation using stack.	-	√	-	-	-
7.	Implement a program to implement queue using list that performs following operations: enqueue, dequeue, display	-	√	-	-	-
8.	Implement program to perform following operation on singly linked list:	-	-	√	-	-

	a. Insert a node at the beginning of a singly linked list. b. Insert a node at the end of a singly linked list. c. Insert a node after the given node of a singly linked list. d. Insert a node before the given node of singly linked list. e. Delete a node from the beginning of a singly linked list. f. Delete a node from the end of a singly linked list. g. Count the number of nodes of a singly linked list. h. Display content of singly linked list					
9.	Implement program to create and display circular linked list.	-	**-**	**√**	-	-
10.	Implement program to perform following operation on doubly linked list: traversal, searching, adding and removing node.	-	-	**√**	-	-
11.	Implement a python program to search a particular element from list using Linear and Binary Search.	-	-	-	**√**	-
12.	Implement Bubble sort algorithm.	-	-	-	**√**	-
13.	Implement Selection sort and Insertion sort algorithm.	-	-	-	**√**	-
14.	Implement Merge sort algorithm.	-	-	-	**√**	**-**
15.	Implement Quick sort algorithm.	-	-	-	**√**	**-**
16.	Implement construction of binary search trees.	-	-	-	-	**√**
17.	Write a menu driven program to perform following operation on Binary Search Tree: a. Create a BST. b. Insert an element in BST. c. Pre-order traversal of BST. d. In-order traversal of BST. e. Post-order traversal of BST. f. Delete an element from BST	-	-	-	-	**√**

Industry Relevant Skills

The following industry relevant skills are expected to be developed in the students by performance of experiments of this course.

Implement various data structures using Python.

Analyze problems, design optimized algorithms, and evaluate their time and space complexity.

Guidelines to Course Faculty

1. Couse faculty should demonstrate experiment with all necessary implementation strategies described in curriculum.
2. Couse faculty should explain industrial relevance before starting of each experiment.
3. Course faculty should involve & give opportunity to all students for hands on experience.
4. Course faculty should ensure mentioned skills are developed in the students by asking.
5. Utilise 2 hrs. of lab hours effectively and ensure completion of write up with quiz also.
6. Encourage peer to peer learning by doing same experiment through fast learners.

Instructions for Students

1. Organize the work in the group and make record of all observations.
2. Students shall develop maintenance skill as expected by industries.
3. Student shall attempt to develop related hand-on skills and build confidence.
4. Student shall develop the habits of evolving more ideas, innovations, skills etc.
5. Student shall refer technical magazines and data books.
6. Student should develop habit to submit the practical on date and time.
7. Student should well prepare while submitting write-up of exercise.

Continuous Assessment Sheet

Enrolment No: **Name:**

Term:

Sr No.	Practical Outcome/Title of experiment	Page	Date	Marks (25)	Sign
1	Design a student class for reading and displaying the student information, the getInfo () and displayInfo () methods will be used respectively. Where getInfo() will be a private method.				
2	Design a class Complex for adding the two complex numbers and show the use of constructor.				
3	Write Python class to represent a bank account. The class contains account holder name, account number, type of account and balance amount as the data members. The member functions are as follows: to initialize the data members, to deposit an amount, to withdraw an amount after checking balance, and to display name and balance.				
4	Write Python class to represent a Cricketer. The class contains name of cricketer, team name and run as the data members. The member functions are as follows: to initialize the data members, to set run and display run.				
5	Implement push and pop algorithms of stack using list.				
6	Implement a program to convert infix notation to postfix notation using stack.				
7	Implement a program to implement queue using list that performs following operations: enqueue, dequeue, display				
8	Implement program to perform following operation on singly linked list: a. Insert a node at the beginning of a singly linked list. b. Insert a node at the end of a singly linked list. c. Insert a node after the given node of a singly linked list. d. Insert a node before the given node of singly linked list.				

	e. Delete a node from the beginning of a singly linked list. f. Delete a node from the end of a singly linked list. g. Count the number of nodes of a singly linked list. h. Display content of singly linked list				
9	Implement program to create and display circular linked list.				
10	Implement program to perform following operation on doubly linked list: traversal, searching, adding and removing node.				
11	Implement a python program to search a particular element from list using Linear and Binary Search.				
12	Implement Bubble sort algorithm.				
13	Implement Selection sort and Insertion sort algorithm.				
14	Implement Merge sort algorithm.				
15	Implement Quick sort algorithm.				
16	Implement construction of binary search trees.				
17	Write a menu driven program to perform following operation on Binary Search Tree: a. Create a BST. b. Insert an element in BST. c. Pre-order traversal of BST. d. In-order traversal of BST. e. Post-order traversal of BST. f. Delete an element from BST				
Total					

Date: ……

Practical No.1: Design an student class for reading and displaying the student information, the getInfo() and displayInfo() methods will be used respectively. Where getInfo() will be a private method.

A. Objective:

To Design a student class with private getInfo() method for reading student information and displayInfo() method for displaying the information.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge Of computing appropriate to the problem.
PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.
PO3:Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to Meet specified needs.
PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified Competency:

1. Object-Oriented Programming (OOP): The practical will help develop skills in designing and implementing classes, encapsulating data and methods, and understanding the concept of private methods.
2. Data Handling and Display: The practical will enhance skills in reading and handling student information, as well as displaying the information in a clear and organized manner. This includes understanding how to access and manipulate data within a class and presenting it appropriately.

D. Expected Course Outcomes(Cos)

CO1: Understand the basic concepts of data structures, analysis terms, OOPs concepts, and comprehend the concept of recursive functions.

E. Practical Outcome(PRo)

Design a student class for reading and displaying the student information, the getInfo() and displayInfo() methods will be used respectively. Where getInfo() will be a private method.

F. Expected Affective domain Outcome(ADos):

1. Increased Sense of Responsibility: The practical will require students to design and implement a class with private methods, which highlights the importance of responsible programming practices. It can cultivate a sense of ownership and responsibility towards maintaining the integrity and security of code.

2. Improved Attention to Detail: The practical involves handling student information and displaying it accurately. Students will need to pay close attention to the data input, processing, and presentation to ensure correctness. This can enhance their attention to detail and promote a thorough approach to programming tasks.

G. **Prerequisite Theory:**

Classes and Objects: A class is a blueprint or template that defines the structure and behaviour of objects. An object is an instance of a class, representing a specific Entity. In the given practical, the student class is designed to represent student Objects.

```
class Student:
    pass

# Creating objects of the Student class
student1 = Student()
student2 = Student()
```

Data Encapsulation and Access Modifiers: Encapsulation is the process of bundling data and methods within a class, providing a level of data protection and hiding implementation details. Access modifiers, such as private and public, control the visibility and accessibility of class members.

```
class Student:
    def __init__(self, name, roll_number):
        self.name = name  # Public attribute
        self.__roll_number = roll_number  # Private attribute

    def displayInfo(self):
        print("Name:", self.name)
        print("Roll Number:", self.__roll_number)

# Creating a student object and accessing its public and private attributes
student = Student("John", 1234)
print(student.name)  # Output: John
print(student.__roll_number)    # Error: 'Student' object has no attribute '__roll_number'
```

Methods: Methods are functions defined within a class and operate on objects of that class. They can perform specific actions or provide functionality related to the class.

```
class Student:
    def __init__(self, name, roll_number):
        self.name = name
        self.roll_number = roll_number

    def displayInfo(self):
        print("Name:", self.name)
        print("Roll Number:", self.roll_number)
# Creating a student object and calling its method
student = Student("John", 1234)
student.displayInfo()  # Output: Name: John, Roll Number: 1234
```

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed:

a. Turn off power switch only after computer is shut down.
b. Do not plug out any computer cables.

J. Procedure to be followed/Source code :

K. Input-Output:

```
>>> s=Student()
>>> s.getInfo()
Enter student's Name: abc
Enter student's roll number: 128383
Enter student's age: 16
>>> s.displayInfo()
Student Information:
Name: abc
Roll No: 128383
Age: 16
```

L. Practical related Quiz.

a. Which method is automatically called when an object of a class is created?

b. How can you define a private method in a class?

c. What is the purpose of encapsulating data and methods within a class?

M. References / Suggestions

Python 3 Object oriented programming by Dusty Phillips, Packt Publishing
https://www.geeksforgeeks.org/python-oops-concepts/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.2: Design a class Complex for adding the two complex numbers and show the use of Constructor.

A. Objective:

To demonstrate the use of a constructor to initialize the complex numbers.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.
PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.
PO3: Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.
PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Acquire a clear understanding of constructors and their role in initializing objects within a class.
2. Develop the ability to translate mathematical concepts, such as complex numbers, into a practical software solution by designing a class that accurately represents and operates on complex numbers.

D. Expected Course Outcomes(Cos)

CO1: Understand the basic concepts of data structures, analysis terms, OOPs concepts,

And comprehend the concept of recursive functions.

E. Practical Outcome(PRo)

Apply knowledge of object-oriented programming and basic mathematics to create a reusable class that performs complex number addition.

F. Expected Affective domain Outcome(ADos)

a. Confidence in Programming: Gain confidence in utilizing object-oriented programming techniques to design and implement classes, contributing to a positive attitude towards programming and its potential for solving engineering problems.

G. Prerequisite Theory:

Complex numbers: Complex numbers are numbers of the form a + bi, where a and b are real numbers and i is the imaginary unit. The real part of the complex number is a, and the imaginary part is b.

To add complex numbers, we add the real parts and the imaginary parts separately. For example, to add the complex numbers z=a+bi and w=c+di, we would add them as follows:

z + w = (a + c) + (b + d)i

Python classes: In Python, classes are used to create user-defined data types. A class is a blueprint for creating objects. Objects are instances of classes.

Constructors: In Python, constructors are special methods that are called when an object is created. The constructor is used to initialize the object's state.

To design a class Complex for adding the two complex numbers, we can use the following steps:

- Define the class Complex. The class should have two attributes: real and imaginary.
- Define the constructor for the class Complex. The constructor should take two arguments: real and imaginary. The constructor should initialize the real and imaginary attributes of the object.
- Define a method for adding two complex numbers. The method should take two complex numbers as arguments. The method should return the sum of the two complex numbers.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

a. Turn off power switch only after computer is shut down.
b. Do not plug out any computer cables

J. Procedure to be followed/Source code :

K. Input-Output:

```
>>> a=Complex(2,3)
>>> b=Complex(5,7)
>>> a.add(b)
'7+10i'
```

L. Practical related Quiz.

a. Design a class Vector for adding and subtracting the two vectors and show the use of constructor.
b. In the "Complex" class, how would you access the real and imaginary parts of a complex number?

M. References / Suggestions (lab manual designer should give)

a. Python 3 Object oriented programming by Dusty Phillips, Packt Publishing
b. https://www.geeksforgeeks.org/python-oops-concepts/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

	identify application of concept.		+very few help.	with few exceptions.		
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.3: Write Python class to represent a bank account. The class contains account holder name, account number, type of account and balance amount as the data members. The member functions are as follows: to initialize the data members, To deposit an amount, to withdraw an amount after checking balance, and to Display name and balance.

A. Objective:

To create a Python class that represents a bank account, allowing users to perform operations such as depositing, withdrawing, and displaying the account holder's name and balance.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge Of computing appropriate to the problem.
PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.
PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes to Meet specified needs.
PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Object-Oriented Programming (OOP): The practical helps develop skills in implementing and utilizing OOP concepts in Python, such as creating a class, defining data members, and implementing member functions.
2. Handling Data and Logic: This practical enhances skills in handling data and implementing logical operations, such as checking balances before allowing withdrawals and displaying relevant information about the bank account.

D. Expected Course Outcomes(Cos)

CO1: Understand the basic concepts of data structures, analysis terms, OOPs concepts, and comprehend the concept of recursive functions.

E. Practical Outcome(PRo)

Create a fully functional Python class that represents a bank account. The class would have the ability to initialize the account holder's name, account number, type of account, and balance amount. It would also provide methods to deposit and withdraw amounts, while ensuring the balance is checked before allowing withdrawals. Additionally, the class would have a method to display the account holder's name and current balance. This practical outcome can be utilized as a foundation for building more complex banking applications or simulations.

F. Expected Affective domain Outcome(ADos)

1. Confidence in Programming: By successfully completing the practical to create a Python class representing a bank account, learners can develop a sense of confidence in their programming skills. They will gain the belief that they can effectively design and implement classes to solve real-world problems.

2. Attention to Detail: The practical requires learners to carefully consider and implement various aspects of the bank account representation, such as data members, member functions, and logical checks. Through this exercise, they can develop an increased attention to detail and the ability to consider different scenarios and edge cases in their programming tasks.

G. Prerequisite Theory:

Classes:

A class is a blueprint or template for creating objects. It defines the attributes and methods that an object of that class will have. The attributes represent the data associated with the object, while the methods define the actions or behaviours that the object can perform.

Example: Let's consider a class called "Car" that represents different types of cars. It can have attributes like "make", "model", and "year".

```
Class Car:
   def. __init__(self, make, model, year):
      self.make = make
      self.model = model
      self.year = year
```

Objects/Instances:

An object, also known as an instance, is a specific realization of a class. It is created using the blueprint provided by the class. Each object has its own set of values for the attributes defined in the class.

Example: We can create instances of the "Car" class to represent different cars.

```
car1 = Car("Toyota", "Camry", 2021)
car2 = Car("Ford", "Mustang", 2019)
```

Methods:

Methods are functions defined within a class and are used to perform operations on the object's attributes or to provide functionality specific to that class.
Example: Let's add a method called "start_engine" to the "Car" class.

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def start_engine(self):
        print("Engine started!")
```

Now, when we create instances of the "Car" class, we can call the "start_engine" method on those objects.

```
car1 = Car("Toyota", "Camry", 2021)
car1.start_engine()  # Output: Engine started!
```

Understanding these fundamental concepts of classes, objects/instances, and methods is crucial for implementing the bank account class. It provides the foundation for creating and manipulating objects representing bank accounts, with attributes like account holder name, account number, type of account, and balance amount, as well as methods to deposit, withdraw, and display information about the account.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

- Turn off power switch only after computer is shut down.
- Do not plug out any computer cables

J. Procedure to be followed/Source code :

K. Input-Output:

```
>>> a=BankAccount('abc',4494959,'savings',50000)
>>> a.display_account_info()
Account Holder Name: abc
Account Number: 4494959
Account Type: savings
Balance: 50000
>>> a.deposit(5000)
Amount 5000 deposited successfully. New balance: 55000
>>> a.withdraw(20000)
Amount 20000 withdrawn successfully. New balance: 35000
>>> a.display_account_info()
Account Holder Name: abc
Account Number: 4494959
Account Type: savings
Balance: 35000
```

L. Practical related Quiz/Exercise.

1. Write a Python class to represent a library book. The class should contain attributes such as book title, author, publication year, and availability status. The member functions should include initializing the data members, updating the availability status, and displaying book details.
2. What is the difference between a class variable and an instance variable? How would you define and access each of them in Python?

M. References / Suggestions (lab manual designer should give)

a. Python 3 Object oriented programming by Dusty Phillips, Packt Publishing

b. https://www.geeksforgeeks.org/python-oops-concepts/

N. Assessment-Rubrics:

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.4: Write Python class to represent a Cricketer. The class contains name of cricketer, team name and run as the data members. The member functions are as follows: To initialize the data members, to set run and display run.

A. Objective:

The objective is to create a Python class that represents a cricketer, with data Members for name, team name, and runs, along with member functions for Initialization, setting runs, and displaying runs.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3: Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

The expected skills to be developed from the practical include proficiency in object-oriented programming, Python programming, data management, problem-solving, code organization, testing and debugging, and understanding class and object concepts.

D. Expected Course Outcomes(Cos)

CO1: Understand the basic concepts of data structures, analysis terms, OOPs concepts, and comprehend the concept of recursive functions.

E. Practical Outcome(PRo)

Use Python class that represents a cricketer, allowing for initialization of data members, setting the runs scored by the cricketer, and displaying the runs.

F. Expected Affective domain Outcome(ADos)

1. Confidence in Programming: By successfully completing the practical to create a Python class representing a bank account, learners can develop a sense of confidence in their programming skills. They will gain the belief that they can effectively design and implement classes to solve real-world problems.
2. Attention to Detail: The practical requires learners to carefully consider and implement various aspects of the bank account representation, such as data members, member functions, and logical checks. Through this exercise, they can develop an increased attention to detail and the ability to consider different scenarios and edge cases in their programming tasks.

G. Prerequisite Theory:

Classes:

A class is a blueprint or template for creating objects. It defines the attributes and methods that an object of that class will have. The attributes represent the data associated with the object, while the methods define the actions or behaviours that the object can perform.

Example: Let's consider a class called "Car" that represents different types of cars. It can have attributes like "make", "model", and "year".

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
```

Objects/Instances:

An object, also known as an instance, is a specific realization of a class. It is created using the blueprint provided by the class. Each object has its own set of values for the attributes defined in the class.

Example: We can create instances of the "Car" class to represent different cars.

```
car1 = Car("Toyota", "Camry", 2021)
car2 = Car("Ford", "Mustang", 2019)
```

Methods:

Methods are functions defined within a class and are used to perform operations on the object's attributes or to provide functionality specific to that class.

Example: Let's add a method called "start_engine" to the "Car" class.

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def start_engine(self):
        print("Engine started!")
```

Now, when we create instances of the "Car" class, we can call the "start_engine" method on those objects.

```
car1 = Car("Toyota", "Camry", 2021)
car1.start_engine()  # Output: Engine started!
```

H. **Resources/Equipment Required :**

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I: Safety and necessary Precautions followed:

(1)Turn off power switch only after computer is shut down.

(2) Do not plug out any computer cables

J: Source code :

K. Input-Output :

```
>>> cricketer1 = Cricketer ("Sachin Tendulkar", "India")

>>> cricketer1.set_run(18426)

>>> cricketer1.display_run()

Sachin Tendulkar has scored 18426 runs.

>>> cricketer2 = Cricketer ("Virat Kohli", "India")

>>> cricketer2.set_run(12169)

>>> cricketer2.display_run()

Virat Kohli has scored 12169 runs.
```

L. Practical related Quiz.

1. Implement a method to update the cricketer's team name and display the updated information.

M. References / Suggestions

a. Introduction to Programming using Python, Y Daniel Liang

b. https://www.geeksforgeeks.org/try-except-else-and-finally-in-python/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

	application of concept.					
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.5: Implement push and pop algorithms of stack using list.

A. Objective:

The objective is to implement the push and pop operations of a stack using a list, aiming to understand and practice the basic operations of a stack data structure.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge Of computing appropriate to the problem.
PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.
PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes to meet specified needs.
PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

.

C. Expected Skills to be developed based on competency:

Based on the practical of implementing push and pop algorithms of a stack using a list, two expected skills to be developed are:

Data Structure Understanding: Implementing the push and pop algorithms of a stack using a list requires a solid understanding of data structures, specifically stacks. This practical will help students develop a deeper understanding of how stacks work, including their basic operations and the concept of LIFO (Last-In, First-Out).

Programming Logic and Implementation: By implementing the push and pop algorithms using a list, students will enhance their programming logic and implementation skills. They will learn how to manipulate data structures, utilize control structures, and manage memory in order to effectively perform stack operations.

D. Expected Course Outcomes(Cos)

CO2: Apply basic operations on stack, queue data structures.

E. Practical Outcome(PRo)

Functional implementation of push and pop algorithms for a stack using a list, demonstrating

Understanding of stack operations and their proper behaviour.

F. Expected Affective domain Outcome(ADos)

Handle computer systems carefully with safety and necessary precaution

Turn off systems after completion of practical lab to save power

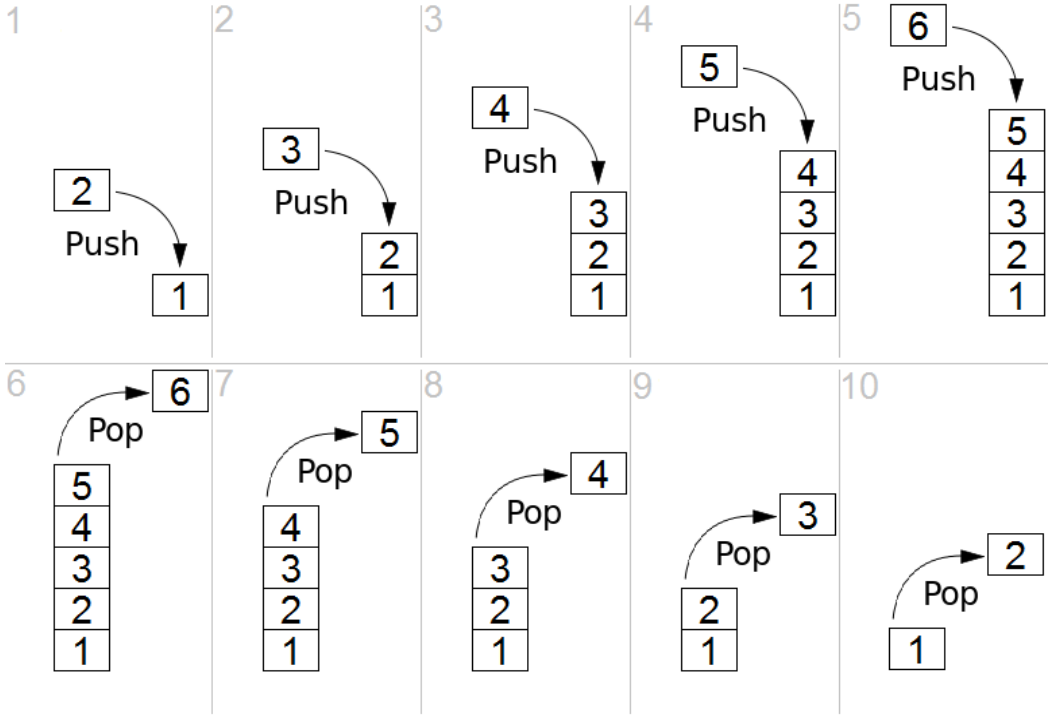
G. Prerequisite Theory:

Stack Data Structure:

A stack is a linear data structure that follows the Last-In, First-Out (LIFO) principle.

It consists of elements arranged in a particular order, where the addition of new elements and the removal of existing elements happen only at one end, called the top.

The top represents the most recently added element, and the bottom represents the least recently added element.



Implementing the Stack using Python List:

The Python list-based implementation of the Stack ADT is the easiest to implement. The first decision we have to make when using the list for the Stack ADT is which end of the list to use as the top and which as the base. For the most efficient ordering, we let the end of the list represent the top of the stack and the front represent the base. As the stack grows, items are appended to the end of the list and when items are popped, they are removed from the same end.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed:

a. *Turn off power switch only after computer is shut down.*
b. *Do not plug out any computer cables*

J. Source code:

K. Input-Output:

```
>>> s=Stack()
>>> s.push(5)
>>> s.push(9)
>>> s.push(6)
>>> s.pop()
6
>>> s.pop()
9
>>> s.pop()
5
>>> s.is_empty()
True
>>> s.size()
0
```

L. Practical related Quiz:

1. *What is the time complexity of the push and pop operations in a stack implemented using a list?*
2. *Can you access or modify items in a stack other than the top item?*
3. *Can a stack contain duplicate elements?*
4. *Is a stack a linear or non-linear data structure?*

M. References / Suggestions (lab manual designer should give)

a. *Data structures and algorithms in python, Goodrich, Wiley publication*
b. https://www.geeksforgeeks.org/stack-in-python/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.6: Implement a program to convert infix notation to postfix notation using stack.

A. Objective:

To demonstrate the conversion of infix notation to postfix notation using a stack data Structure, illustrating the concept of operator precedence and associativity in Expressions.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge Of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.

PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Data Structures and Algorithms: Understanding and implementing the stack data structure efficiently to manage the conversion process.
2. Problem-Solving: Analyzing complex expressions and developing algorithms to convert infix to postfix notation, showcasing problem-solving abilities.
3. Programming Proficiency: Strengthening programming skills in the chosen language, as the implementation involves manipulating strings and data structures.

D. Expected Course Outcomes(Cos)

CO2: Apply basic operations on stack, queue data structures.

E. Practical Outcome(PRo)

A functional program that successfully converts infix notation expressions to postfix notation using a stack data structure, demonstrating the understanding and application of operator precedence and associativity rules in expressions.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution
2. Enhancing self-confidence and self-efficacy in programming and problem-solving.

G. Prerequisite Theory:

Infix expression: The expression of the form “a operator b” (a + b) i.e., when an operator is in-between every pair of operands.

Postfix expression: The expression of the form “a b operator” (ab+) i.e., When every pair of operands is followed by an operator.

How to convert an Infix expression to a Postfix expression?
To convert infix expression to postfix expression, use the stack data structure. Scan the infix expression from left to right. Whenever we get an operand, add it to the postfix expression and if we get an operator or parenthesis add it to the stack by maintaining their precedence.

Below are the steps to implement the above idea:

1. Scan the infix expression from left to right.
2. If the scanned character is an operand, put it in the postfix expression.
3. Otherwise, do the following
   - If the precedence and associativity of the scanned operator are greater than the precedence and associativity of the operator in the stack [or the stack is empty or the stack contains a ‘(‘], then push it in the stack. [‘^‘operator is right associative and other operators like ‘+‘,’–‘,’*‘and ‘/‘are left-associative].
   - Check especially for a condition when the operator at the top of the stack and the scanned operator both are ‘^‘. In this condition, the precedence of the scanned operator is higher due to its right associativity. So it will be pushed into the operator stack.
   - In all the other cases when the top of the operator stack is the same as the scanned operator, then pop the operator from the stack because of left associativity due to which the scanned operator has less precedence.
   - Else, pop all the operators from the stack which are greater than or equal to in precedence than that of the scanned operator.
   - After doing that push the scanned operator to the stack. (If you encounter parenthesis while popping then stop there and push the scanned operator in the stack.)
4. If the scanned character is a ‘(‘, push it to the stack.
5. If the scanned character is a ‘)’, pop the stack and output it until a ‘(‘is encountered, and discard both the parenthesis.
6. Repeat steps 2-5 until the infix expression is scanned.
7. Once the scanning is over, Pop the stack and add the operators in the postfix expression until it is not empty.
8. Finally, print the postfix expression.

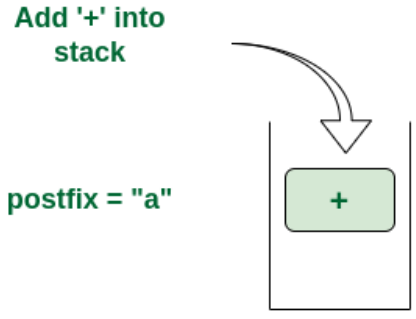
Illustration:
Follow the below illustration for a better understanding

Consider the infix expression exp = “a+b*c+d”
and the infix expression is scanned using the iterator i, which is initialized as i = 0.

1st Step: Here i = 0 and exp[i] = ‘a’ i.e., an operand. So add this in the postfix expression. Therefore, postfix = “a”.

'a' is an operand. Add it in postfix expression

2nd Step: Here i = 1 and exp[i] = ‘+’ i.e., an operator. Push this into the stack. postfix = “a” and stack = {+}.

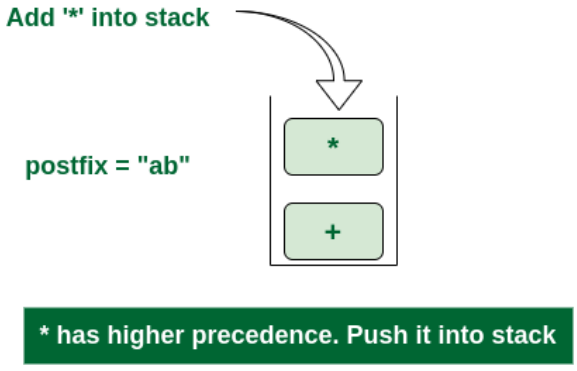


Stack is empty. Push '+' into stack

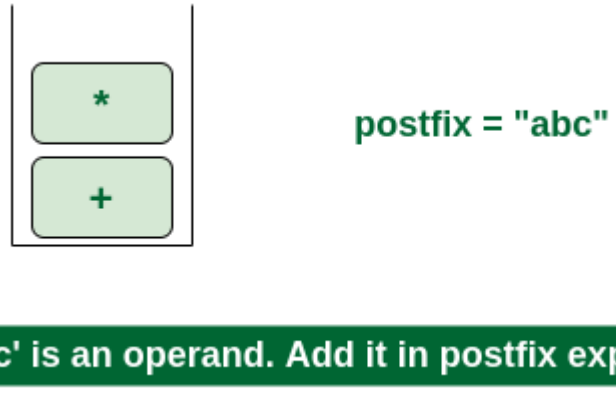
3rd Step: Now i = 2 and exp[i] = ‘b’ i.e., an operand. So add this in the postfix expression. postfix = “ab” and stack = {+}.

'd' is an operand. Add it in postfix expression

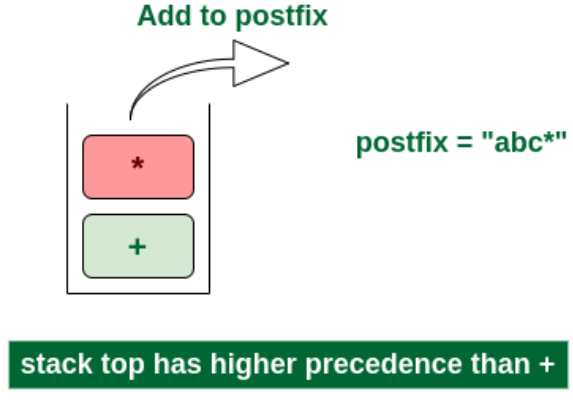
4th Step: Now i = 3 and exp[i] = '*' i.e., an operator. Push this into the stack. postfix = "ab" and stack = {+, *}.



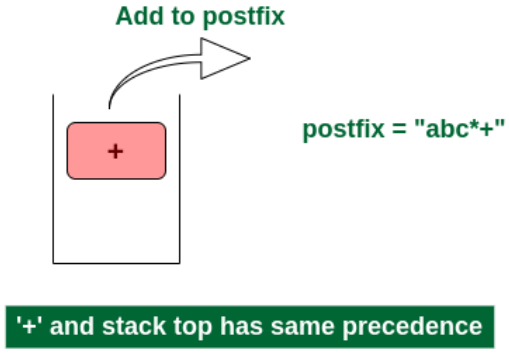
5th Step: Now i = 4 and exp[i] = 'c' i.e., an operand. Add this in the postfix expression. postfix = "abc" and stack = {+, *}.



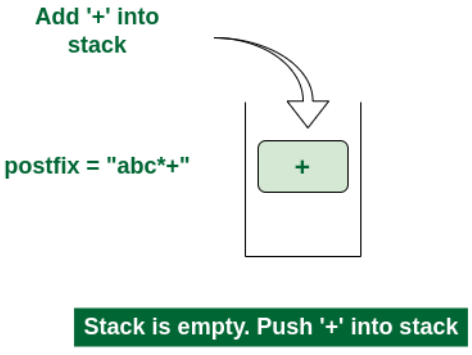
6th Step: Now i = 5 and exp[i] = '+' i.e., an operator. The topmost element of the stack has higher precedence. So pop until the stack becomes empty or the top element has less precedence. '*' is popped and added in postfix. So postfix = "abc*" and stack = {+}.



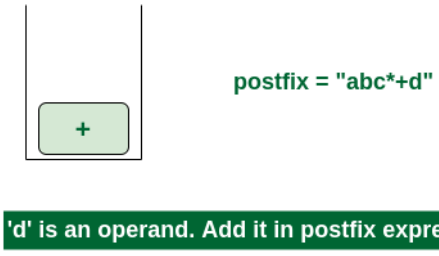
Now top element is '+' that also doesn't have less precedence. Pop it. postfix = "abc*+".



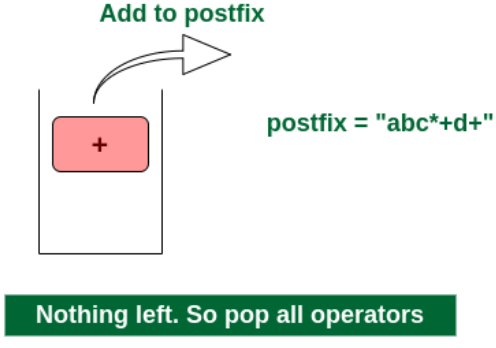
Now stack is empty. So push '+' in the stack. stack = {+}.



7th Step: Now i = 6 and exp[i] = 'd' i.e., an operand. Add this in the postfix expression. postfix = "abc*+d".



Final Step: Now no element is left. So empty the stack and add it in the postfix expression. postfix = "abc*+d+".



H. Resources/Equipment Required

Sr. No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output :

Enter the infix expression: 3 + 4 * (2 - 1)
Postfix expression: 3 4 2 1 - * +

L. Practical related Quiz.

1. Convert the following infix expressions to postfix notation using the stack-based algorithm:
   a) A * (B + C) - D / (E + F)
   b) 9 / 3 + 5 * (7 - 2)
   c) 12 - (4 * 3 + 2) / 6

M. References / Suggestions (lab manual designer should give)

1. Data structures and algorithms in python, Goodrich, Wiley publication
2. https://www.geeksforgeeks.org/convert-infix-expression-to-postfix-expression/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified

Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.7: Implement a program to implement queue using list that performs following operations: enqueue, dequeue, display.

A. Objective:

To implement a program that uses a list to simulate a queue and perform operations such as enqueue, dequeue, and display.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3:Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Understanding the concept of queues and their implementation using lists.
2. Problem-solving skills to handle edge cases and ensure the correct functioning of the queue Implementation.
3. Effective coding practices, such as modular and readable code, proper variable naming, and Code documentation.

D. Expected Course Outcomes(Cos)

CO2: Apply basic operations on stack, queue data structures.

E. Practical Outcome(PRo)

A working program that implements a queue using a list, allowing for enqueue, dequeue, and display operations.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution
2. Enhancing self-confidence and self-efficacy in programming and problem-solving.

G. Prerequisite Theory:

A queue is a specialized list with a limited number of operations in which items can only be added to one end and removed from the other. A queue is also known as a first in first-out (FIFO) list. Consider Figure below, which illustrates an abstract

View of a queue. New items are inserted into a queue at the back while existing items are removed from the front. Even though the illustration shows the individual items, they cannot be accessed directly. The deletion of the Queue ADT follows.

A queue is a data structure that a linear collection of items in which access is restricted to a first-in first-out basis. New items are inserted at the back and existing items are removed from the front. The items are maintained in the order in which they are added to the structure.

_ Queue (): Creates a new empty queue, which is a queue containing no items.
_ isEmpty(): Returns a boolean value indicating whether the queue is empty.
_ length (): Returns the number of items currently in the queue.
_ enqueue(item): Adds the given item to the back of the queue.
_ dequeue(): Removes and returns the front item from the queue. An item can-not be dequeued from an empty queue.

H. Resources/Equipment Required

Sr. No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output :

```
>>> q=Queue()
>>> q.enqueue(5)
Enqueued: 5
>>> q.display()
Queue elements: [5]
>>> q.enqueue(7)
Enqueued: 7
>>> q.enqueue(9)
Enqueued: 9
>>> q.display()
Queue elements: [5, 7, 9]
>>> q.dequeue()
Dequeued: 5
5
>>> q.display()
Queue elements: [7, 9]
>>> q.dequeue()
```

```
Dequeued: 7
7
>>> q.dequeue()
Dequeued: 9
9
>>> q.dequeue()
Queue is empty. Cannot dequeue.
>>> q.is_empty()
True.
```

L. Practical related Quiz.

1. Which data structure is used to implement a queue in the provided program?

Answer:

2. Can you enqueue items of different data types in the queue?

Answer:

M. References / Suggestions (lab manual designer should give)

3. Data structures and algorithms in python, Goodrich, Wiley publication
4. https://www.geeksforgeeks.org/queue-in-python/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.8:Implement program to perform following operation on singly linked list:

a. Insert a node at the beginning of a singly linked list.
b. Insert a node at the end of a singly linked list.
c. Insert a node after the given node of a singly linked list.
d. Insert a node before the given node of singly linked list.
e. Delete a node from the beginning of a singly linked list.
f. Delete a node from the end of a singly linked list.
g. Count the number of nodes of a singly linked list.
h. Display content of singly linked list

A. Objective:

To implement a program that can perform operations like inserting nodes at the beginning, end, or after/before a given node, deleting nodes from the beginning or end, counting the number of nodes, and displaying the content of a singly linked list.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge Of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.

PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Understanding of Linked List Data Structure: Gain a clear understanding of the concept and implementation of a singly linked list data structure.
2. Node Manipulation: Learn how to insert and delete nodes at different positions in a linked list, such as at the beginning, end, after, or before a given node.

D. Expected Course Outcomes(Cos)

CO3: Apply basic operations on linked list data structure.

E. Practical Outcome(PRo)

a) Gain hands-on experience in manipulating singly linked lists by implementing Operations like node insertion, deletion, and traversal.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution
2. Turn off systems after completion of practical lab to save power

G. Prerequisite Theory:

1. Singly Linked List:

- A singly linked list is a linear data structure composed of nodes, where each node contains data and a pointer/reference to the next node in the list.

- The first node in the list is called the head, and the last node is called the tail. The tail node's pointer/reference is typically set to `None` to indicate the end of the list.

- Traversing a singly linked list can only be done in one direction, starting from the head and moving towards the tail.

2. Node:

- A node is a fundamental building block of a linked list. It contains two components:

  - Data: The actual value or information stored in the node.

  - Next Pointer/Reference: A pointer/reference that points to the next node in the list.

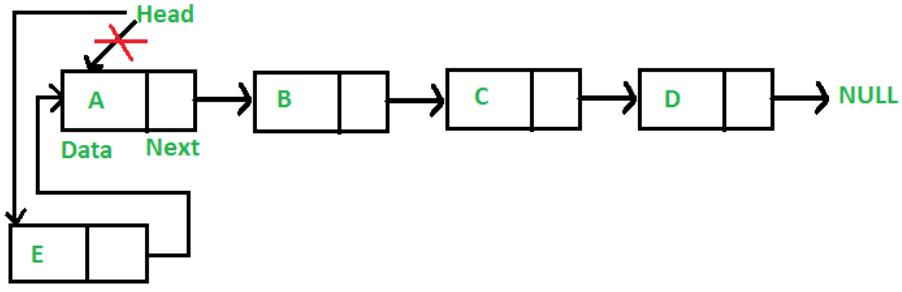
3. Operations on a Singly Linked List:

a. Insertion at the Beginning:

- To insert a node at the beginning of a linked list, create a new node with the desired data.

- Set the next pointer of the new node to the current head of the list.

- Update the head pointer to point to the new node.

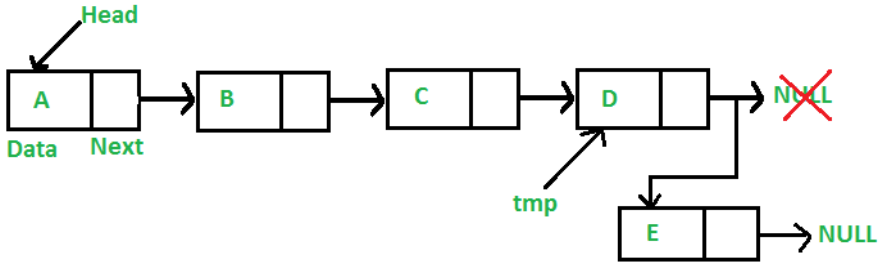


b. Insertion at the End:

- To insert a node at the end of a linked list, create a new node with the desired data.

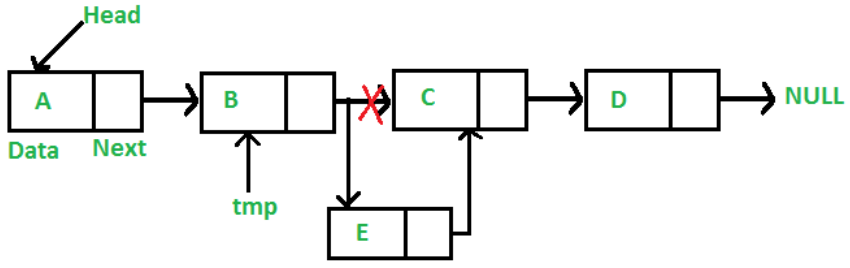
- If the list is empty, set the new node as both the head and the tail.

- Otherwise, set the next pointer of the current tail to the new node and update the tail pointer to the new node.



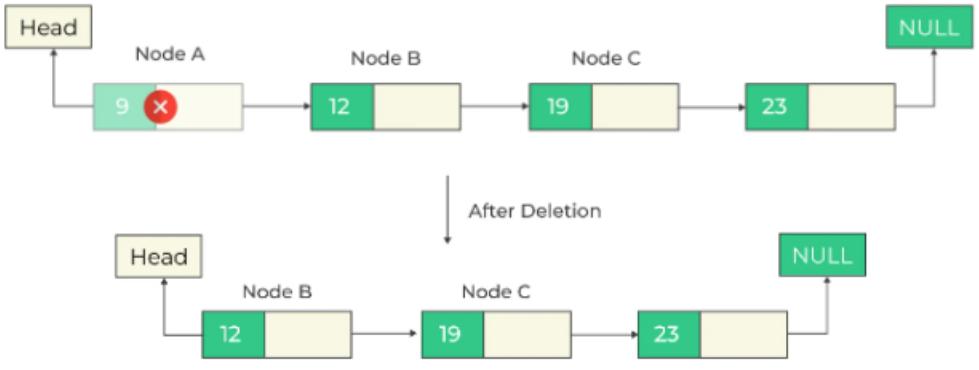
c. Insertion after a Given Node:

- To insert a node after a given node, create a new node with the desired data.
- Set the next pointer of the new node to the next node of the given node.
- Update the next pointer of the given node to point to the new node.



d. Deletion from the Beginning:

- To delete the first node of a linked list, update the head pointer to point to the next node.
- Optionally, free the memory occupied by the deleted node.

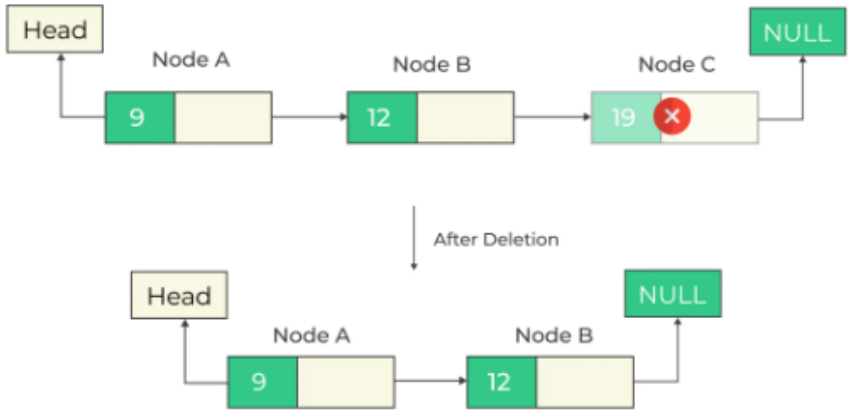


e. Deletion from the End:

- To delete the last node of a linked list, find the second-to-last node (node before the tail).

- Set the next pointer of the second-to-last node to `None`, indicating the new tail.

- Optionally, free the memory occupied by the deleted node.



f. Counting Nodes:

- To count the number of nodes in a linked list, start from the head node and traverse the list while incrementing a counter variable.

- Keep moving to the next node until reaching the end (when the next pointer is `None`).

- Return the final count.

g. Displaying Linked List Content:

- To display the content of a linked list, start from the head node and traverse the list.

- Access and print the data of each node.

- Optionally, use formatting techniques (e.g., commas or arrows) to separate the data for better readability.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

(1)

```
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
>>> a.display()
7 -> None
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
```

(2)

```
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
>>> a.display()
7 -> None
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
>>> a.insert_at_beginning(9)
>>> a.display()
9 -> 8 -> 7 -> None
>>> a.insert_at_end(10)
```

```
>>> a.display()
9 -> 8 -> 7 -> 10 -> None

(3)
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
>>> a.insert_after_node(8,9)
>>> a.display()
8 -> 9 -> 7 -> None
>>> a.insert_after_node(7,10)
>>> a.display()
8 -> 9 -> 7 -> 10 -> None

(4)
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
>>> a.insert_after_node(8,9)
>>> a.display()
8 -> 9 -> 7 -> None
>>> a.insert_before_node(7,10)
>>> a.display()
8 -> 9 ->10-> 7 -> None

(5)
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
>>> a.insert_after_node(8,9)
>>> a.display()
8 -> 9 -> 7 -> None
>>> a.delete_from_beginning()
>>> a.display()
 9 -> 7 -> None
(6)
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
```

```
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
>>> a.insert_after_node(8,9)
>>> a.display()
8 -> 9 -> 7 -> None
>>> a.delete_from_end()
>>> a.display()
 8 -> 9 -> None
```

(7)

```
>>> a=SinglyLinkedList()
>>> a.insert_at_beginning(7)
>>> a.insert_at_beginning(8)
>>> a.display()
8 -> 7 -> None
>>> a.insert_after_node(8,9)
>>> a.display()
8 -> 9 -> 7 -> None
>>> a.count_nodes()
3
```

L. Practical related Quiz.

1. Explain the steps to insert a node at the end of a singly linked list.
2. Explain the process of deleting a node from the end of a singly linked list.
3. What is the time complexity of inserting a node at the beginning, end, and after a given node in the singly linked list?

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.geeksforgeeks.org/data-structures/linked-list/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

	identify application of concept.					
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.9: Implement program to create and display circular linked list.

A. Objective:

To implement a Python program to create and display a circular linked list. This data structure consists of nodes where each node points to the next node in the list, forming a circular loop, with the last node pointing back to the first node. The program aims to illustrate the concept of circular linked lists and their traversal for display.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.

PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Linked List Operations: Learners will become proficient in performing common linked list operations, such as creating nodes, linking them, and traversing the list.
2. Understanding Circular Data Structures: By working with circular linked lists, participants will deepen their understanding of circular data structures and their applications.

D. Expected Course Outcomes(Cos)

CO3: Apply basic operations on linked list data structure.

E. Practical Outcome(PRo)

a) The practical outcome is a Python program that demonstrates the creation and display of a circular linked list, showcasing the ability to manage node connections and traverse the circular structure.

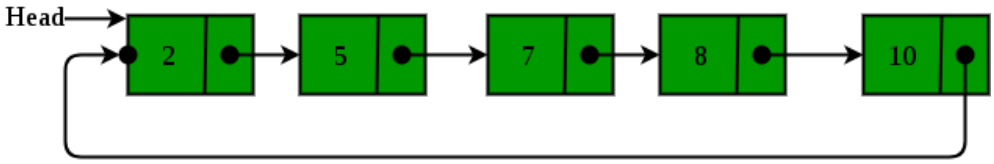
F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

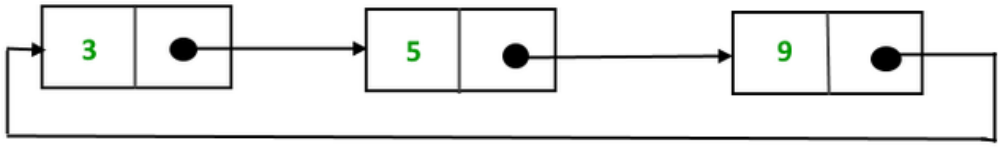
What is Circular linked list?

The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which forms a circle. There is no NULL at the end.



There are generally two types of circular linked lists:

Circular singly linked list: In a circular singly linked list, the last node of the list contains a pointer to the first node of the list. We traverse the circular singly linked list until we reach the same node where we started. The circular singly linked list has no beginning or end. No null value is present in the next part of any of the nodes.



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

```
>>> a=CircularLinkedList()
>>> a.insert_at_beginning(8)
>>> a.insert_at_beginning(9)
>>> a.insert_at_beginning(10)
>>> a.display()
10 -> 9 -> 8 ->
```

L. Practical related Quiz.

1. What is a Circular Linked List, and how does it differ from a traditional singly linked list?
2. Explain the process of traversing a Circular Linked List and displaying its elements.
3. Write a function to delete a specific node from a Circular Linked List based on its data value.

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.geeksforgeeks.org/circular-linked-list/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.10: Implement program to perform following operation on doubly linked list : traversal, searching, adding, and removing node.

A. Objective:

To develop a program that demonstrates the basic operations of traversal, searching, adding, and removing nodes in a doubly linked list, showcasing proficiency in linked list manipulation.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3:Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4:Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

A. Proficiency in Data Structure Manipulation: By completing the practical on doubly linked list operations, participants can develop essential skills in working with data structures, particularly linked lists. This competency is crucial in the industry, as linked lists are fundamental components of various software applications and systems.

B. Problem-Solving and Algorithmic Thinking: Designing and implementing functions for traversal, searching, adding, and removing nodes in a doubly linked list require strong problem-solving and algorithmic thinking skills. Through this practical, participants can enhance their ability to analyze problems, devise efficient solutions, and implement them correctly, which is highly valuable in real-world programming scenarios.

D. Expected Course Outcomes(Cos)

CO3: Apply basic operations on linked list data structure.

E. Practical Outcome(PRo)

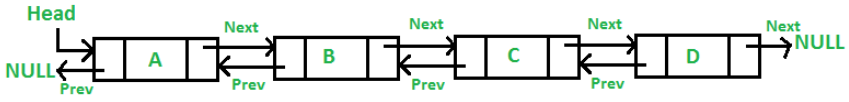
a) Practical outcome of this practical is a fully functional program that demonstrates the implementation of traversal, searching, adding, and removing nodes in a doubly linked list.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

A Doubly Linked List (DLL) contains an extra pointer, typically called the previous pointer, together with the next pointer and data which are there in a singly linked list.



Below are operations on the given DLL:

Add a node at the front of DLL: The new node is always added before the head of the given Linked List. And the newly added node becomes the new head of DLL & maintaining a global variable for counting the total number of nodes at that time.

Traversal of a Doubly linked list

Insertion of a node: This can be done in three ways:

At the beginning: The new created node is insert in before the head node and head points to the new node.

At the end: The new created node is insert at the end of the list and tail points to the new node.

At a given position: Traverse the given DLL to that position(let the node be X) then do the following:

- Change the next pointer of new Node to the next pointer of Node X.
- Change the prev pointer of next Node of Node X to the new Node.
- Change the next pointer of node X to new Node.
- Change the prev pointer of new Node to the Node X.

Deletion of a node: This can be done in three ways:

At the beginning: Move head to the next node to delete the node at the beginning and make previous pointer of current head to NULL .

At the last: Move tail to the previous node to delete the node at the end and make next pointer of tail node to NULL.

At a given position: Let the prev node of Node at position pos be Node X and next node be Node Y, then do the following:

Change the next pointer of Node X to Node Y.

Change the previous pointer of Node Y to Node X.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

```
>>> a=DoublyLinkedList()
>>> a.add(5)
>>> a.add(7)
>>> a.add(4)
>>> a.display_forward()
5 7 4
>>> a.display_backward()
4 7 5
>>> a.search(7)
```

```
True
>>> a.search(10)
False
>>> a.remove(7)
True
>>> a.display_forward()
5 4
```

L. Practical related Quiz.

1. What is a doubly linked list, and how does it differ from a singly linked list?
2. Describe the process of searching for a specific element in a doubly linked list.
3. Explain the steps to add a new node after a specific node in a doubly linked list.

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.geeksforgeeks.org/doubly-linked-list-tutorial/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.11: Implement a python program to search a particular element from list using Linear and Binary Search.

A. Objective:

To compare and understand the differences in the efficiency and implementation of linear search and binary search algorithms for finding a specific element in a list.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3:Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4:Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Algorithm Optimization: The practical on linear and binary search helps develop the skill of optimizing algorithms to improve search efficiency, which is essential in industries that deal with large datasets and performance-critical applications.
2. Problem Solving and Analysis: By working on these search algorithms, learners can enhance their problem-solving and analytical skills, which are highly valued in various industries, including software development, data analysis, and research. These skills enable professionals to approach complex problems systematically and find effective solutions.

D. Expected Course Outcomes(Cos)

CO4: Apply different sorting and searching algorithms to the small data sets.

E. Practical Outcome(PRo)

a. Students will gain proficiency in selecting and implementing appropriate search algorithms (linear or binary) based on the problem requirements, leading to more efficient and effective searching of elements in lists.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Types of Search Algorithms:

- Linear or Sequential Search
- Binary Search

Linear or Sequential Search

This algorithm works by sequentially iterating through the whole array or list from one end until the target element is found. If the element is found, it returns its index, else -1.

Now let's look at an example and try to understand how it works:

arr = [2, 12, 15, 11, 7, 19, 45]

Suppose the target element we want to search is 7.

Approach for Linear or Sequential Search

- Start with index 0 and compare each element with the target
- If the target is found to be equal to the element, return its index
- If the target is not found, return -1

Time Complexity Analysis

The Best Case occurs when the target element is the first element of the array. The number of comparisons, in this case, is 1. So, the time complexity is O(1).

The Average Case: On average, the target element will be somewhere in the middle of the array. The number of comparisons, in this case, will be N/2. So, the time complexity will be O(N) (the constant being ignored).

The Worst Case occurs when the target element is the last element in the array or not in the array. In this case, we have to traverse the entire array, and so the number of comparisons will be N. So, the time complexity will be O(N).

Binary Search

This type of searching algorithm is used to find the position of a specific value contained in a sorted array. The binary search algorithm works on the principle of divide and conquer and it is considered the best searching algorithm because it's faster to run.

Now let's take a sorted array as an example and try to understand how it works:

arr = [2, 12, 15, 17, 27, 29, 45]

Suppose the target element to be searched is 17.

Approach for Binary Search

- Compare the target element with the middle element of the array.
- If the target element is greater than the middle element, then the search continues in the right half.
- Else if the target element is less than the middle value, the search continues in the left half.

- This process is repeated until the middle element is equal to the target element, or the target element is not in the array
- If the target element is found, its index is returned, else -1 is returned.

Time Complexity Analysis

The Best Case occurs when the target element is the middle element of the array. The number of comparisons, in this case, is 1. So, the time complexity is O(1).

The Average Case: On average, the target element will be somewhere in the array. So, the time complexity will be O(logN).

The Worst Case occurs when the target element is not in the list or it is away from the middle element. So, the time complexity will be O(logN).

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

given list: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
Enter element you want to search:14
Linear Search: Element 14 found at index 6
Binary Search: Element 14 found at index 6

L. Practical related Quiz.

1. When is linear search preferred over binary search?
2. Explain how binary search works and what condition is required for it to work correctly.
3. When should binary search be used, and what are its limitations?

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.freecodecamp.org/news/search-algorithms-linear-and-binary-search-explained/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.12: Implement Bubble sort algorithm.

A. Objective:

To implement the Bubble Sort algorithm in Python. Bubble Sort is a simple sorting algorithm that repeatedly compares adjacent elements and swaps them if they are in the wrong order until the entire list is sorted.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3:Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4:Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Algorithm Implementation and Optimization: Developing skills in implementing and optimizing sorting algorithms like Bubble Sort to improve efficiency and execution time.
2. Problem-Solving and Critical Thinking: Enhancing critical thinking abilities to analyze problems, choose appropriate algorithms, and apply them effectively in real-world scenarios.

D. Expected Course Outcomes(Cos)

CO4: Apply different sorting and searching algorithms to the small data sets.

E. Practical Outcome(PRo)

a) Successful implementation of the Bubble Sort algorithm in Python, providing a basic understanding of sorting algorithms and their application.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Bubble Sort Algorithm

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

In this algorithm,

traverse from left and compare adjacent elements and the higher one is placed at right side.

In this way, the largest element is moved to the rightmost end at first.

This process is then continued to find the second largest and place it and so on until the data is sorted.

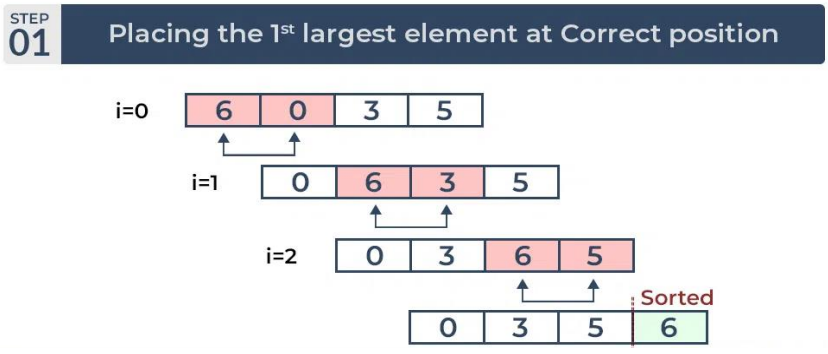
How does Bubble Sort Work?

Let us understand the working of bubble sort with the help of the following illustration:

Input: list=[6, 3, 0, 5]

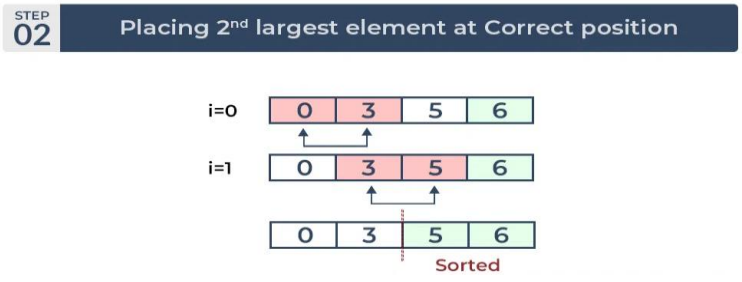
First Pass:

The largest element is placed in its correct position, i.e., the end of the array.



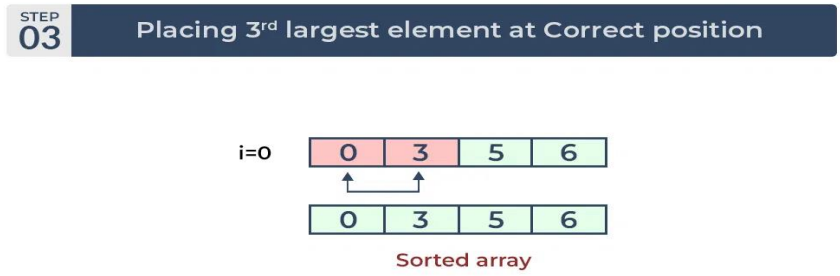
Second Pass:

Place the second largest element at correct position



Third Pass:

Place the remaining two elements at their correct positions.



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

Unsorted list: [64, 34, 25, 12, 22, 11, 90]
Sorted list: [11, 12, 22, 25, 34, 64, 90]

L. Practical related Quiz.

1. Explain how the Bubble Sort algorithm works step by step with an example array.
2. Discuss the advantages and disadvantages of using Bubble Sort compared to other sorting algorithms in various scenarios.

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.freecodecamp.org/news/search-algorithms-linear-and-binary-search-explained/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification

Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.13: Implement Selection sort and Insertion sort algorithm.

A. Objective:

To implement Selection Sort and Insertion Sort algorithms in Python to gain a better understanding of sorting techniques, practice Python programming, and become familiar with the time complexities of these algorithms.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.

PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Algorithm Implementation and Optimization: Being able to implement and understand sorting algorithms like Selection Sort and Insertion Sort demonstrates competency in algorithm implementation, and further optimization of these algorithms (or implementing more advanced sorting algorithms) is an important skill in industries dealing with large datasets and performance-critical applications.
2. Problem-Solving and Critical Thinking: The practical involves solving sorting problems using Python, which enhances problem-solving skills and encourages critical thinking in terms of selecting the appropriate algorithms and optimizing code efficiency, both of which are highly valued in various industries, including software development, data analysis, and machine learning.

D. Expected Course Outcomes(Cos)

CO4: Apply different sorting and searching algorithms to the small data sets.

E. Practical Outcome(PRo)

a) Successful implementation of Selection Sort and Insertion Sort algorithms in Python, providing working solutions for sorting arrays and demonstrating competency in basic sorting techniques and Python programming.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.

2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Selection sort:

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.

The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

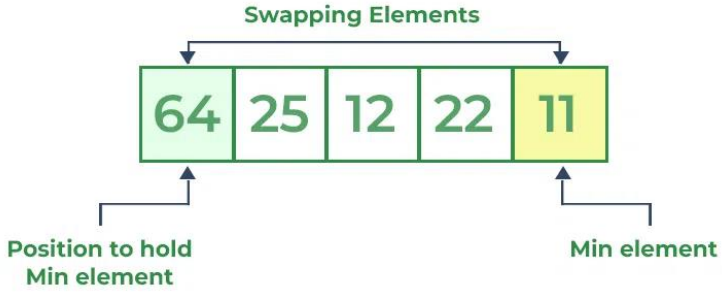
How does Selection Sort Algorithm work?

Lets consider the following array as an example: arr = [64, 25, 12, 22, 11]

First pass:

For the first position in the sorted array, the whole array is traversed from index 0 to 4 sequentially. The first position where 64 is stored presently, after traversing whole array it is clear that 11 is the lowest value.

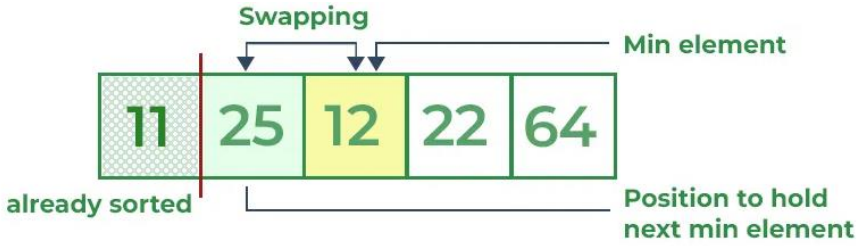
Thus, replace 64 with 11. After one iteration 11, which happens to be the least value in the array, tends to appear in the first position of the sorted list.



Second Pass:

For the second position, where 25 is present, again traverse the rest of the array in a sequential manner.

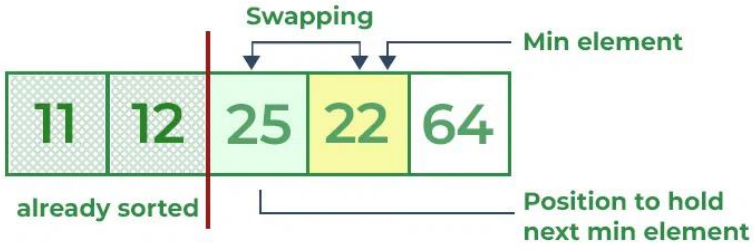
After traversing, we found that 12 is the second lowest value in the array and it should appear at the second place in the array, thus swap these values.



Third Pass:

Now, for third place, where 25 is present again traverse the rest of the array and find the third least value present in the array.

While traversing, 22 came out to be the third least value and it should appear at the third place in the array, thus swap 22 with element present at third position.



Fourth pass:

Similarly, for fourth position traverse the rest of the array and find the fourth least element in the array

As 25 is the 4th lowest value hence, it will place at the fourth position.

Fifth Pass:

At last the largest value present in the array automatically get placed at the last position in the array

The resulted array is the sorted array.

Insertion sort:

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

Insertion Sort Algorithm

To sort an array of size N in ascending order iterate over the array and compare the current element (key) to its predecessor, if the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

Working of Insertion Sort algorithm

Consider an example: arr= [12, 11, 13, 5, 6]

First Pass:

Initially, the first two elements of the array are compared in insertion sort.

Here, 12 is greater than 11 hence they are not in the ascending order and 12 is not at its correct position. Thus, swap 11 and 12.

So, for now 11 is stored in a sorted sub-array.

Second Pass:

Now, move to the next two elements and compare them

Here, 13 is greater than 12, thus both elements seems to be in ascending order, hence, no swapping will occur. 12 also stored in a sorted sub-array along with 11

Third Pass:

Now, two elements are present in the sorted sub-array which are 11 and 12

Moving forward to the next two elements which are 13 and 5

Both 5 and 13 are not present at their correct place so swap them

After swapping, elements 12 and 5 are not sorted, thus swap again

Here, again 11 and 5 are not sorted, hence swap again

Here, 5 is at its correct position

Fourth Pass:

Now, the elements which are present in the sorted sub-array are 5, 11 and 12

Moving to the next two elements 13 and 6

Clearly, they are not sorted, thus perform swap between both

Finally, the array is completely sorted.

H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	Computer system with operating system		1
2.	Python IDLE		1

I. Safety and necessary Precautions followed

1. Turn off power switch only after computer is shut down.
2. Do not plug out any computer cables

J. Source code :

K. Input-Output:

(1) Selection sort

Unsorted list: [64, 34, 25, 12, 22, 11, 90]

Sorted list [11, 12, 22, 25, 34, 64, 90]

(2) Insertion sort

Unsorted list: [64, 34, 25, 12, 22, 11, 90]

Sorted list [11, 12, 22, 25, 34, 64, 90]

L. Practical related Quiz.

1. Explain how the Bubble Sort algorithm works step by step with an example array.
2. Discuss the advantages and disadvantages of using Bubble Sort compared to other sorting algorithms in various scenarios.

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication

https://www.geeksforgeeks.org/selection-sort/

https://www.geeksforgeeks.org/insertion-sort/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.14: Implement Merge sort algorithm.

A. Objective:

To implement the Merge sort algorithm in Python and understand its efficient sorting technique for a given list of elements..

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3:Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4:Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Algorithm Implementation: The practical on implementing the Merge sort algorithm in Python helps develop the skill of effectively implementing complex algorithms, which is crucial in various industries where efficient sorting and data manipulation are required.
2. Problem Solving and Optimization: By working on the Merge sort algorithm, students can enhance their problem-solving abilities and learn about optimization techniques to improve the performance of sorting algorithms, which is valuable in real-world scenarios where large datasets need to be sorted efficiently.

D. Expected Course Outcomes(COs)

CO4: Apply different sorting and searching algorithms to the small data sets.

E. Practical Outcome(PRo)

a) A fully functional and efficient Python sorting function using Merge sort, enhancing algorithm design and problem-solving skills..

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Merge sort:

Merge sort is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array.

In simple terms, we can say that the process of merge sort is to divide the array into two halves, sort each half, and then merge the sorted halves back together. This process is repeated until the entire array is sorted.

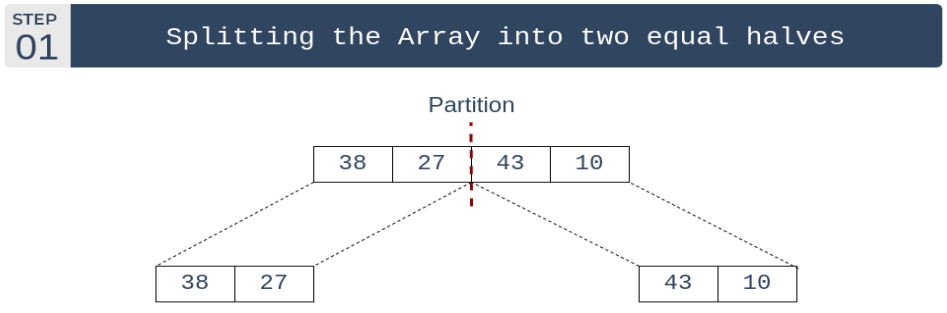
How does Merge Sort work?

Merge sort is a recursive algorithm that continuously splits the array in half until it cannot be further divided i.e., the array has only one element left (an array with one element is always sorted). Then the sorted subarrays are merged into one sorted array.

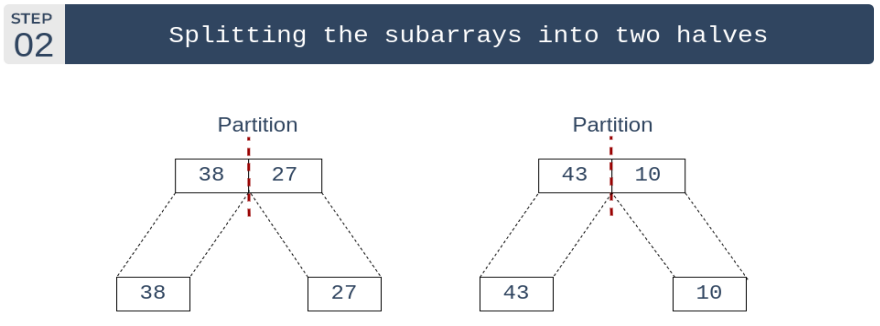
Illustration:

Lets consider an array arr= [38, 27, 43, 10]

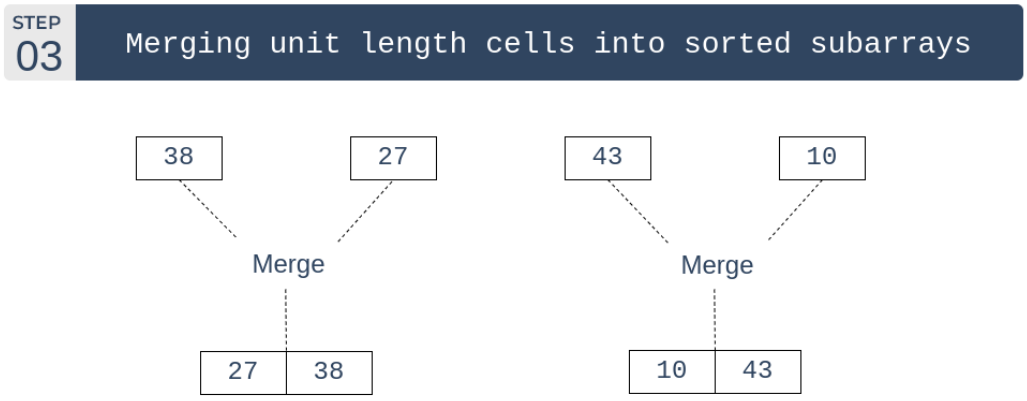
Initially divide the array into two equal halves:



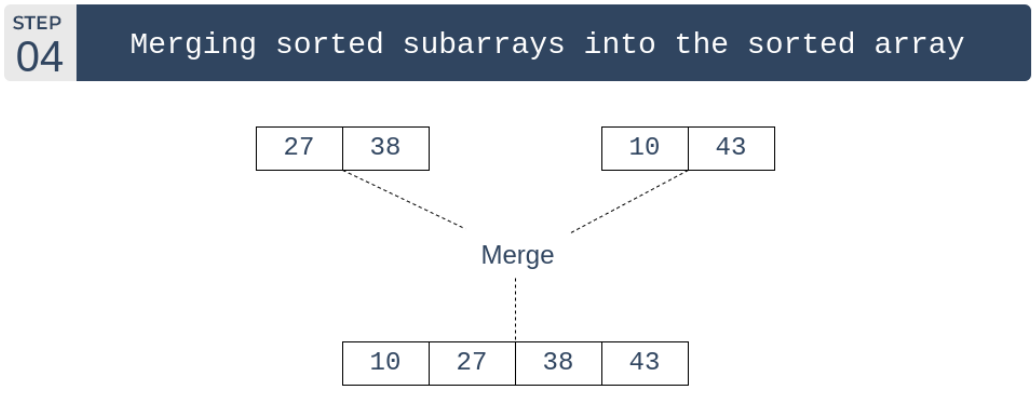
These sub arrays are further divided into two halves. Now they become array of unit length that can no longer be divided and array of unit length are always sorted.



These sorted sub arrays are merged together, and we get bigger sorted sub arrays.



This merging process is continued until the sorted array is built from the smaller sub arrays.



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

Unsorted list: [38, 27, 43, 3, 9, 82, 10]
Sorted list: [3, 9, 10, 27, 38, 43, 82]

L. Practical related Quiz.

1. Explain the divide-and-conquer approach used in Merge sort.
2. What are the advantages of using Merge sort over other sorting algorithms?
3. What is the time complexity of Merge sort, and why is it considered efficient?

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.geeksforgeeks.org/merge-sort/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.15: Implement Quick sort algorithm.

A. Objective:

To implement the Quick sort algorithm in Python, gaining understanding of its inner workings, practicing programming skills, and optimizing sorting efficiency using the divide-and-conquer technique.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3:Design/ development of solutions: Design solutions for *engineering* well-defined technical problems and assist with the design of systems components or processes to meet specified needs.

PO4:Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools and appropriate technique to conduct standard tests and measurements..

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Algorithmic Problem Solving: The ability to understand, design, and implement efficient algorithms like Quick sort is highly valued in the industry. This skill is crucial for optimizing various processes, data manipulation, and handling complex tasks effectively.

D. Expected Course Outcomes(Cos)

CO4: Apply different sorting and searching algorithms to the small data sets.

E. Practical Outcome(PRo)

- Students' ability to implement the Quick sort algorithm in Python, understand its inner workings, optimize code efficiency, and gain practical experience in applying the divide-and-conquer technique for solving sorting problems in real-world scenarios.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Quick sort:

QuickSort is a sorting algorithm based on the Divide and Conquer algorithm that picks an element as a pivot and partitions the given array around the picked pivot by placing the pivot in its correct position in the sorted array.

How does Merge Sort work?

The key process in quickSort is a partition(). The target of partitions is to place the pivot (any element can be chosen to be a pivot) at its correct position in the sorted array and put all smaller elements to the left of the pivot, and all greater elements to the right of the pivot.

Partition is done recursively on each side of the pivot after the pivot is placed in its correct position and this finally sorts the array.

Choice of Pivot:

There are many different choices for picking pivots.

- Always pick the first element as a pivot.
- Always pick the last element as a pivot (implemented below)
- Pick a random element as a pivot.
- Pick the middle as the pivot.

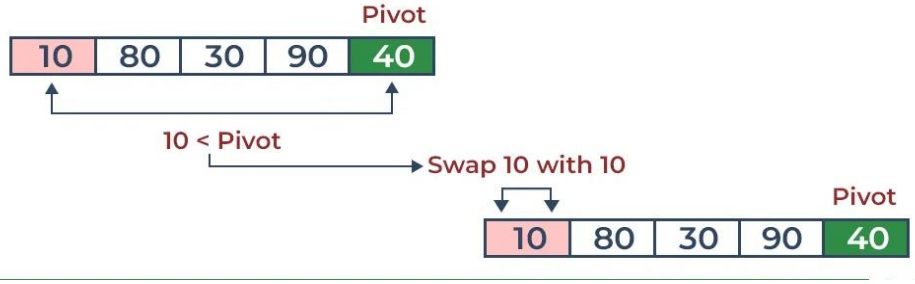
Partition Algorithm:

The logic is simple, we start from the leftmost element and keep track of the index of smaller (or equal) elements as i. While traversing, if we find a smaller element, we swap the current element with arr[i]. Otherwise, we ignore the current element.

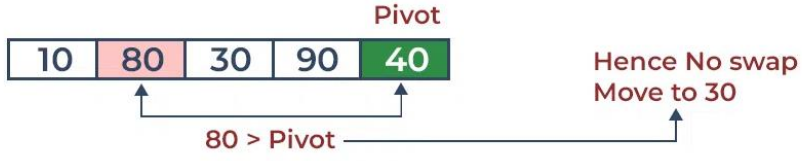
Let us understand the working of partition and the Quick Sort algorithm with the help of the following example:

Consider: arr= [10, 80, 30, 90, 40]

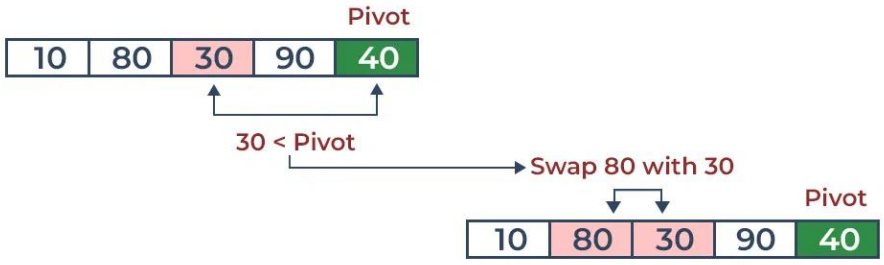
Compare 10 with the pivot and as it is less than pivot arrange it accrodingly.



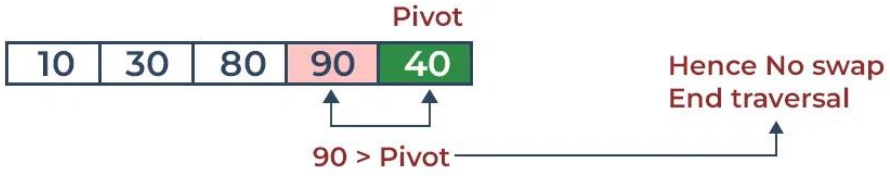
Compare 80 with the pivot. It is greater than pivot.



Compare 30 with pivot. It is less than pivot so arrange it accordingly.



Compare 90 with the pivot. It is greater than the pivot.



Arrange the pivot in its correct position.

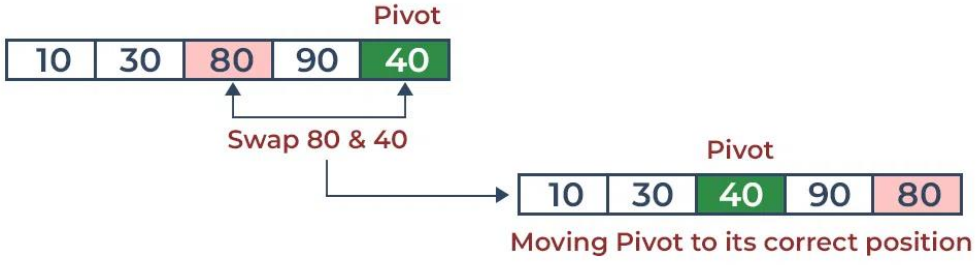
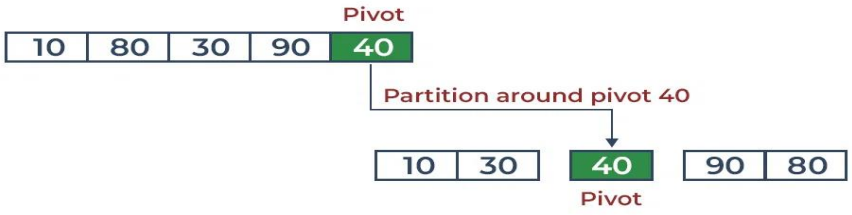


Illustration of Quicksort:

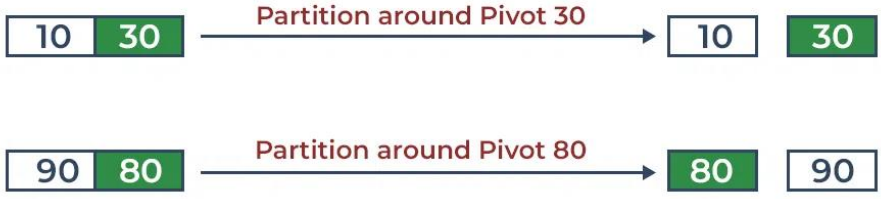
As the partition process is done recursively, it keeps on putting the pivot in its actual position in the sorted array. Repeatedly putting pivots in their actual position makes the array sorted.

Follow the below images to understand how the recursive implementation of the partition algorithm helps to sort the array.

Initial partition on the main array:



Partitioning of the subarrays:



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

Unsorted List: [7, 2, 1, 6, 8, 5, 3, 4]
Sorted List: [1, 2, 3, 4, 5, 6, 7, 8]

L. Practical related Quiz.

1. Explain the divide-and-conquer technique and how it is applied in the Quick sort algorithm.
2. Analyze the time complexity of Quick sort and compare it with other sorting algorithms like Merge sort and Bubble sort.
3. Discuss the advantages and limitations of Quick sort.

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.geeksforgeeks.org/quick-sort/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.

Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.16: Implement construction of binary search trees.

A. **Objective:**To implement the construction of a binary search tree in Python, enabling efficient data insertion and retrieval operations.

B. **Expected Program Outcomes (POs)**

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge of computing appropriate to the problem.

PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using codified standard methods.

PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.

PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. **Expected Skills to be developed based on competency:**

This practical is expected to develop the following skills for the industry-identified competency:

- Data Structures and Algorithms Proficiency: The practical helps in understanding and applying the concepts of binary search trees, which are fundamental data structures in computer science. This competency would enable professionals to design and implement efficient algorithms for various problem-solving tasks.
- Problem-Solving and Optimization: Building and working with binary search trees requires critical thinking and problem-solving skills to ensure the tree is balanced and performs optimally. This competency would be valuable in various industries where efficient data storage and retrieval are essential, such as software development, database management, and information systems.

D. **Expected Course Outcomes(Cos)**

CO5: Illustrate algorithms to insert, delete and searching a node in tree.

E. **Practical Outcome(PRo)**

- Students' ability to implement the Quick sort algorithm in Python, understand its inner workings, optimize code efficiency, and gain practical experience in applying the divide-and-conquer technique for solving sorting problems in real-world scenarios.

F. **Expected Affective domain Outcome(ADos)**

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Binary Search Tree:

A Binary Search Tree (BST) is a type of binary tree data structure that follows a specific rule for organizing its nodes. In a binary tree, each node can have at most two children: a left child and a right child. The BST property ensures that for every node in the tree, all nodes in its left subtree have values less than or equal to the node's value, and all nodes in its right subtree have values greater than the node's value.

The main characteristics of a Binary Search Tree are:

BST Property: As mentioned earlier, for any node N in the tree:

- All nodes in N's left sub tree have values less than or equal to the value of N.
- All nodes in N's right sub tree have values greater than the value of N.

Unique Key: Each node in the BST has a unique key (value). No two nodes can have the same key.

Recursive Structure: The BST is defined recursively, as each sub tree is also a valid BST.

The BST property allows for efficient search, insertion, and deletion operations. When searching for a value in a BST, the tree can be traversed efficiently by comparing the target value with the values at each node and choosing the appropriate sub tree to continue the search. This property also ensures that inserting a new node into the tree can be done in a way that maintains the BST structure.

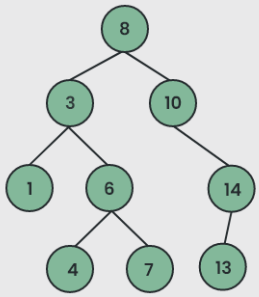
Algorithm to search for a key in a given Binary Search Tree:

Let’s say we want to search for the number X, We start at the root. Then:

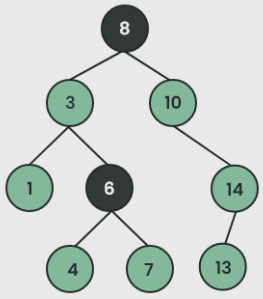
- We compare the value to be searched with the value of the root.
- If it’s equal we are done with the search if it’s smaller we know that we need to go to the left sub tree because in a binary search tree all the elements in the left sub tree are smaller and all the elements in the right sub tree are larger.
- Repeat the above step till no more traversal is possible
- If at any iteration, key is found, return true. Else False.

Illustration of searching in a BST:

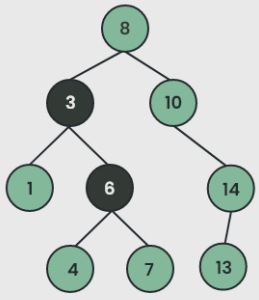
See the illustration below for a better understanding:



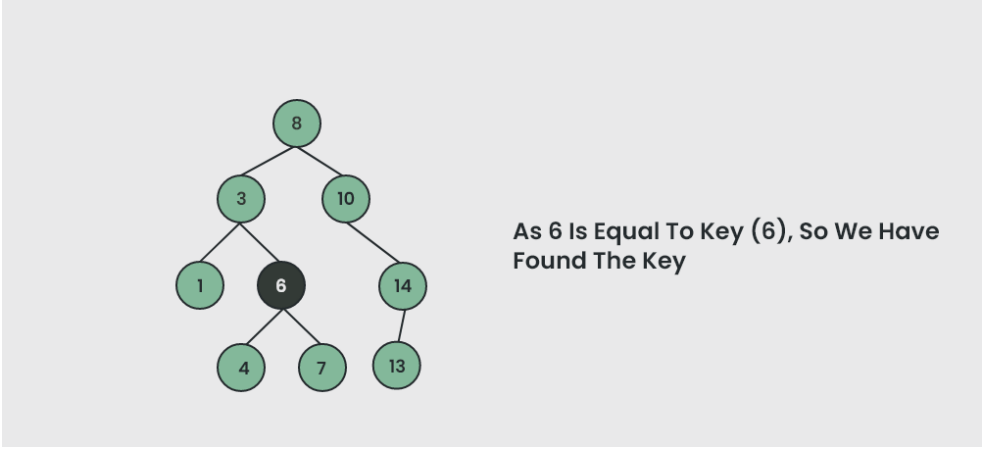
Consider The Following BST
Key = 6



Compare Key With Root, i.e 8
as 6<8, search in left subtree
of 8



As Key (6) Is Greater Than 3,
Search In The Right Subtree Of 3



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quantity
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

Building tree with these elements: [17, 4, 1, 20, 9, 23, 18, 34]
In order traversal of tree : [1, 4, 9, 17, 18, 20, 23, 34]

L. Practical related Quiz.

1. Explain the concept of the "binary search property" in a binary search tree.
2. What are the three common methods to traverse a binary search tree?
3. How do you define the height of a binary search tree?

M. References / Suggestions (lab manual designer should give)

- Data structures and algorithms in python, Goodrich, Wiley publication
- https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented

Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date

Date: ……

Practical No.17: Write a menu driven program to perform following operation on Binary Search Tree:

a. Create a BST.
b. Insert an element in BST.
c. Pre-order traversal of BST.
d. In-order traversal of BST.
e. Post-order traversal of BST.
f. Delete an element from BST.

A. Objective:

To implement a program that allows users to perform various operations on a Binary Search Tree (BST). These operations include creating a BST, inserting elements into the BST, performing different types of tree traversals (pre-order, in-order, post-order), and deleting elements from the BST. The program provides an interactive way for users to interact with and manipulate the BST data structure.

B. Expected Program Outcomes (POs)

PO1: Basic and Discipline specific knowledge: Be able to apply engineering knowledge Of computing appropriate to the problem.
PO2: Problem analysis: Identify and analyse well-defined *engineering* problems using Codified standard methods.
PO3: Design/ development of solutions: Design solutions for *engineering* well-defined Technical problems and assist with the design of systems components or processes To meet specified needs.
PO4: Engineering Tools, Experimentation and Testing: Apply modern *engineering* tools And appropriate technique to conduct standard tests and measurements.

C. Expected Skills to be developed based on competency:

This practical is expected to develop the following skills for the industry-identified competency:

1. Data Structure and Algorithm Proficiency: Understanding how to implement and work with data structures like Binary Search Trees is a crucial skill in the software development industry. Knowledge of various data structures, their properties, and their operations allows developers to make informed decisions about selecting the right data structure for a specific problem and efficiently managing data.

2. Problem-Solving and Software Design: The practical involves creating a menu-driven program to perform operations on a Binary Search Tree, which requires problem-solving and software design skills. Being able to break down a problem into smaller components, design a modular and maintainable solution, and implement it effectively are vital skills for any software

engineer. This practical helps in honing these skills through the design and implementation of the BST operations and menu-driven user interface.

D. Expected Course Outcomes(Cos)

CO5: Illustrate algorithms to insert, delete and search a node in a tree.

E. Practical Outcome(PRo)

a) Students will create a program that allows users to create and manipulate Binary Search Trees through insertion, deletion, and various traversal methods.

F. Expected Affective domain Outcome(ADos)

1. Handle computer systems carefully with safety and necessary precaution.
2. Turn off systems after completion of practical lab to save power.

G. Prerequisite Theory:

Binary Search Tree:

A Binary Search Tree (BST) is a type of binary tree data structure that follows a specific rule for organizing its nodes. In a binary tree, each node can have at most two children: a left child and a right child. The BST property ensures that for every node in the tree, all nodes in its left sub tree have values less than or equal to the node's value, and all nodes in its right sub tree have values greater than the node's value.

The main characteristics of a Binary Search Tree are:

BST Property: As mentioned earlier, for any node N in the tree:

- All nodes in N's left sub tree have values less than or equal to the value of N.
- All nodes in N's right sub tree have values greater than the value of N.

Unique Key: Each node in the BST has a unique key (value). No two nodes can have the same key.

Recursive Structure: The BST is defined recursively, as each sub tree is also a valid BST.

The BST property allows for efficient search, insertion, and deletion operations. When searching for a value in a BST, the tree can be traversed efficiently by comparing the target value with the values at each node and choosing the appropriate sub tree to continue the search. This property also ensures that inserting a new node into the tree can be done in a way that maintains the BST structure.

Algorithm to search for a key in a given Binary Search Tree:

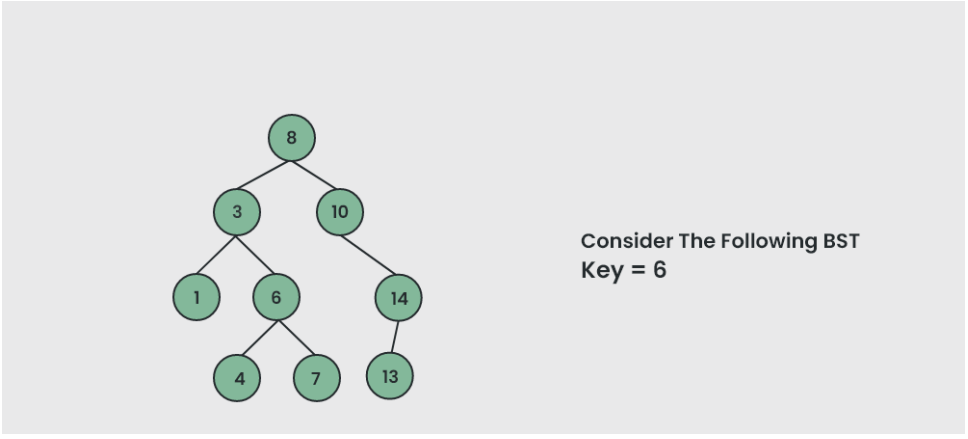
Let’s say we want to search for the number X, We start at the root. Then:

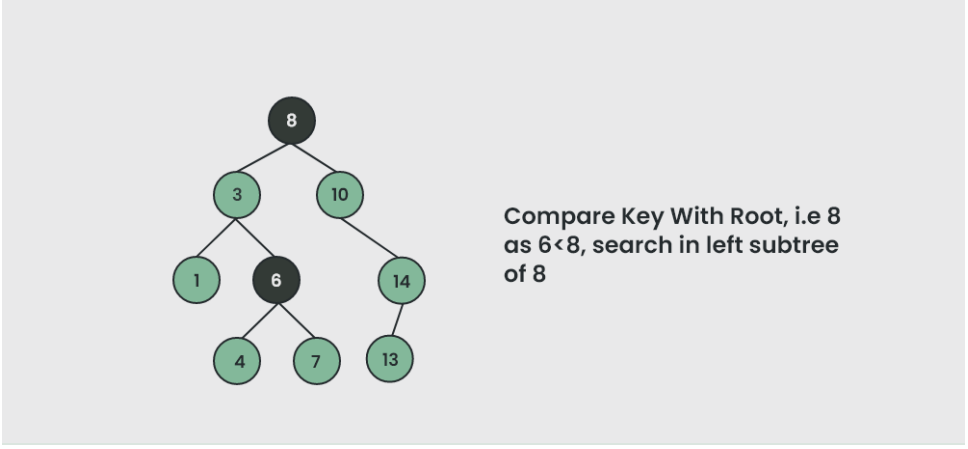
- We compare the value to be searched with the value of the root.

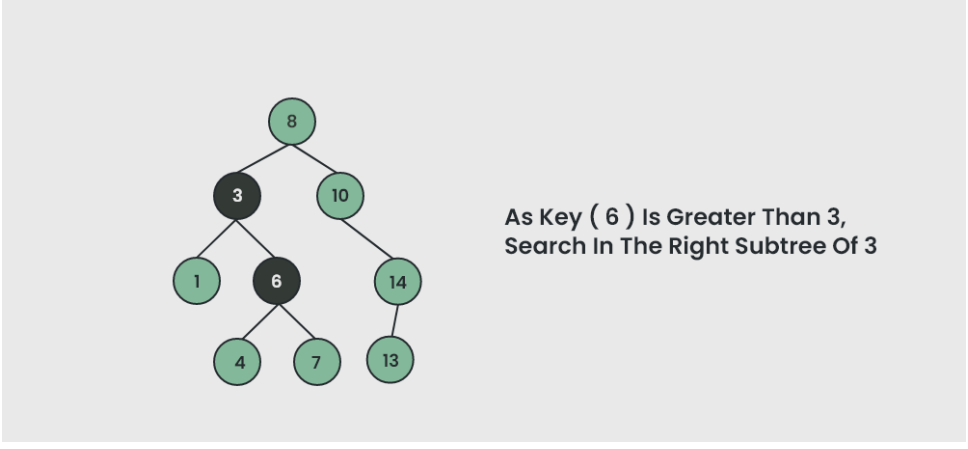
- If it's equal we are done with the search if it's smaller we know that we need to go to the left sub tree because in a binary search tree all the elements in the left sub tree are smaller and all the elements in the right subtree are larger.
- Repeat the above step till no more traversal is possible
- If at any iteration, key is found, return True. Else False.

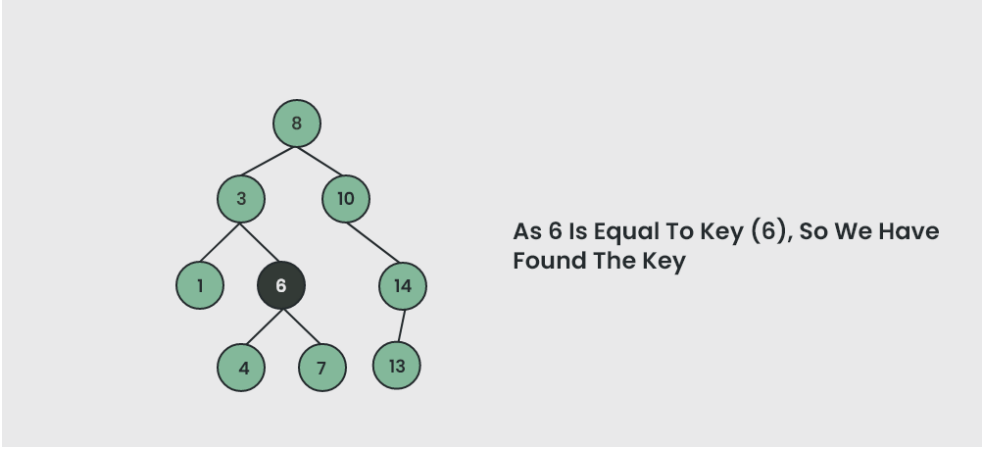
Illustration of searching in a BST:

See the illustration below for a better understanding:



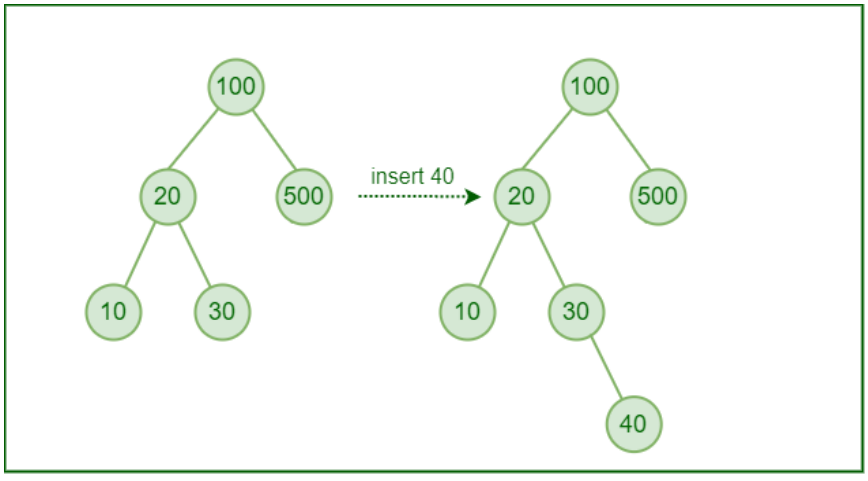






- **Insertion in Binary Search Tree (BST):**

  Given a BST, the task is to insert a new node in this BST.



How to insert a value in a Binary Search Tree:

A new key is always inserted at the leaf by maintaining the property of the binary search tree. We start searching for a key from the root until we hit a leaf node. Once a leaf node is found, the new node is added as a child of the leaf node. The below steps are followed while we try to insert a node into a binary search tree:

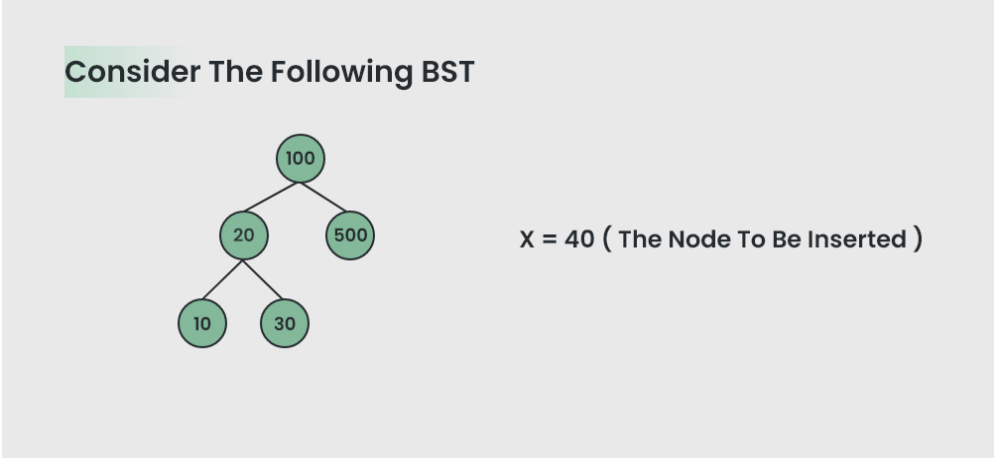
Check the value to be inserted (say X) with the value of the current node (say val) we are in:

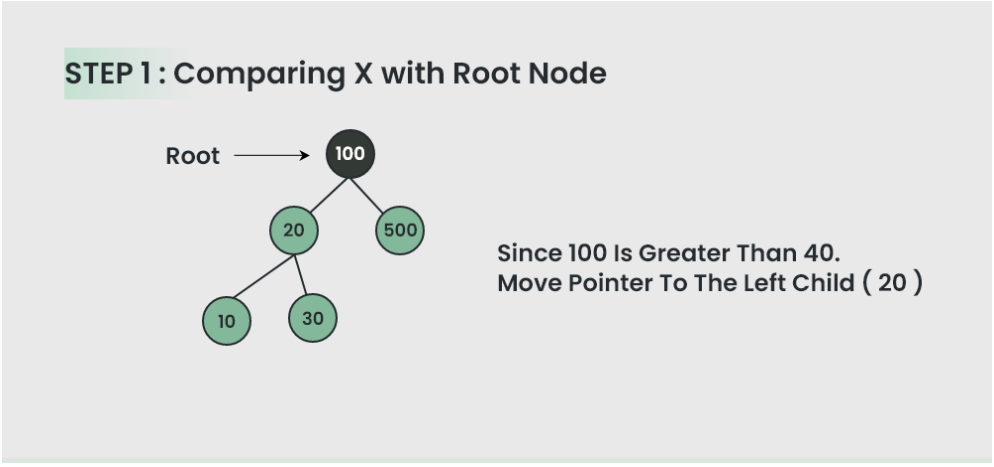
If X is less than val move to the left subtree.

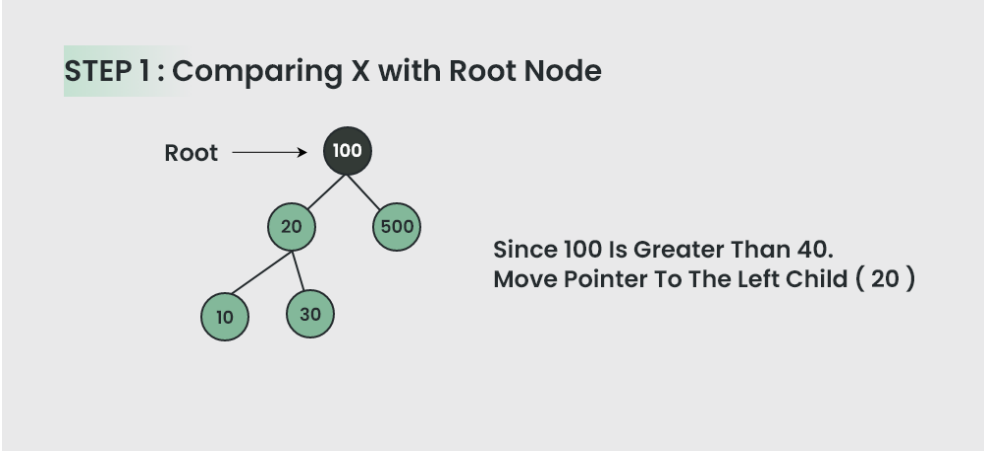
Otherwise, move to the right subtree.

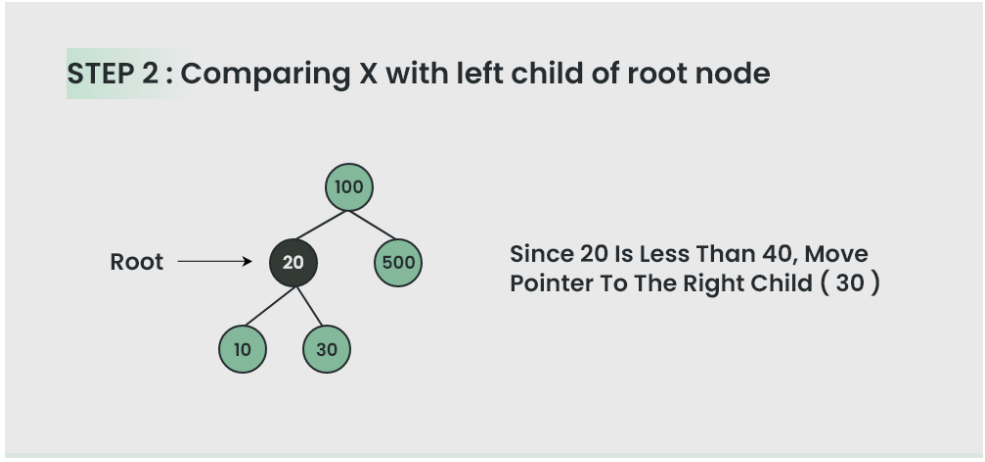
Once the leaf node is reached, insert X to its right or left based on the relation between X and the leaf node's value.

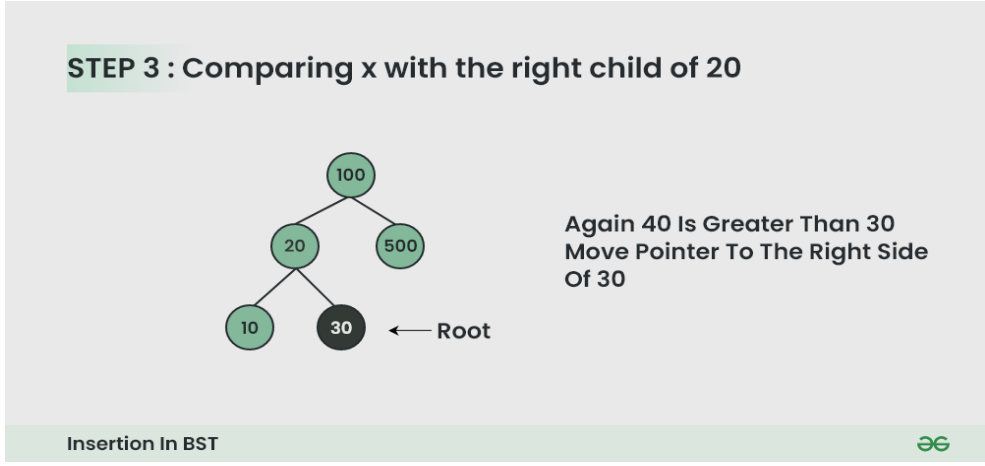
Follow the below illustration for a better understanding:

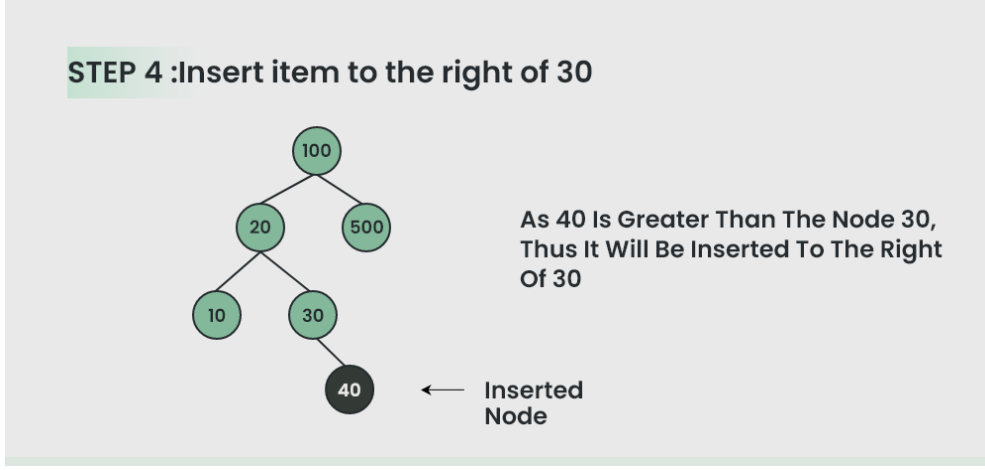




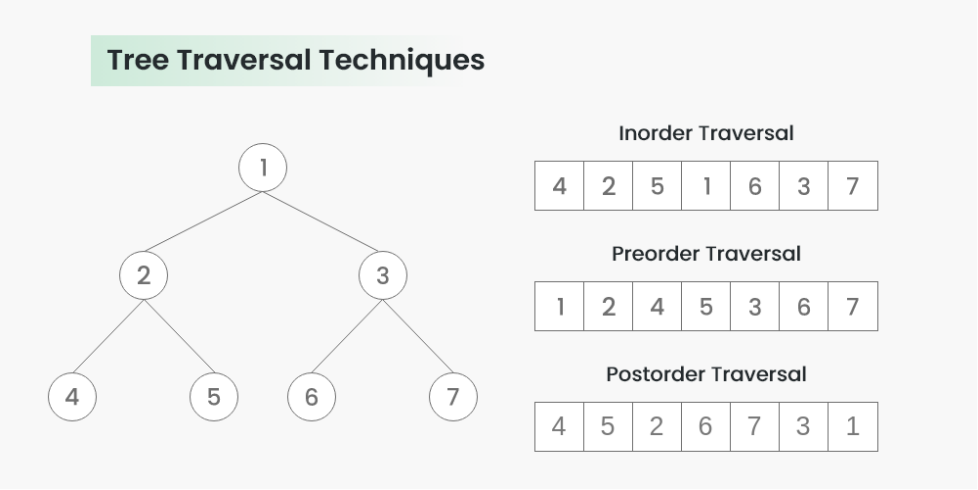



STEP 2 : Comparing X with left child of root node
100
Root
20
500
10
30
Since 20 Is Less Than 40, Move Pointer To The Right Child (30)


STEP 3 : Comparing x with the right child of 20
100
20
500
10
30
Root
Again 40 Is Greater Than 30 Move Pointer To The Right Side Of 30
Insertion In BST


STEP 4 :Insert item to the right of 30
100
20
500
10
30
40
Inserted Node
As 40 Is Greater Than The Node 30, Thus It Will Be Inserted To The Right Of 30

- **BST Traversal Techniques:**



1. Inorder Traversal:

Algorithm Inorder(tree)

- Traverse the left subtree, i.e., call Inorder(left->subtree)
- Visit the root.
- Traverse the right subtree, i.e., call Inorder(right->subtree)

Uses of Inorder Traversal:

In the case of binary search trees (BST), Inorder traversal gives nodes in non-decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal is reversed can be used.

2. Preorder Traversal:

Algorithm Preorder(tree)

- Visit the root.
- Traverse the left subtree, i.e., call Preorder(left->subtree)
- Traverse the right subtree, i.e., call Preorder(right->subtree)

Uses of Preorder:

Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expressions on an expression tree.

3. Postorder Traversal:

Algorithm Postorder(tree)

- Traverse the left subtree, i.e., call Postorder(left->subtree)

- Traverse the right subtree, i.e., call Postorder(right->subtree)
- Visit the root

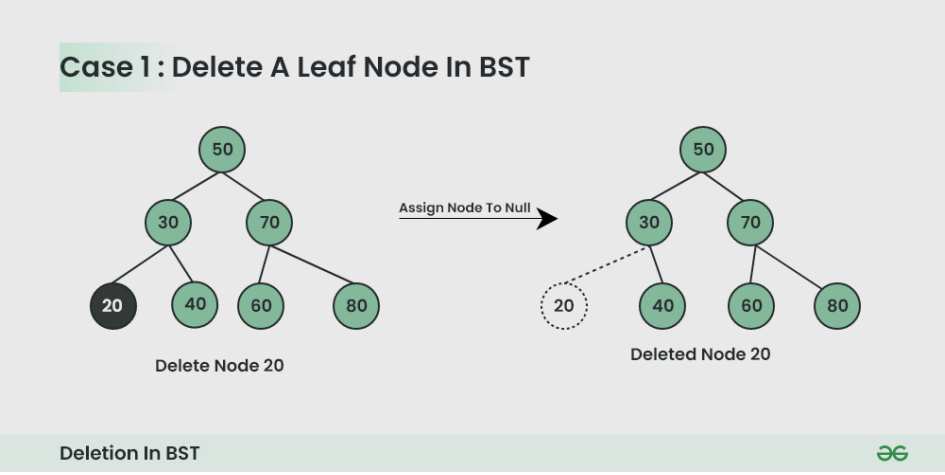
Uses of Postorder:

Postorder traversal is used to delete the tree. Please see the question for the deletion of a tree for details. Postorder traversal is also useful to get the postfix expression of an expression tree.

- **Deletion in Binary Search Tree (BST):**

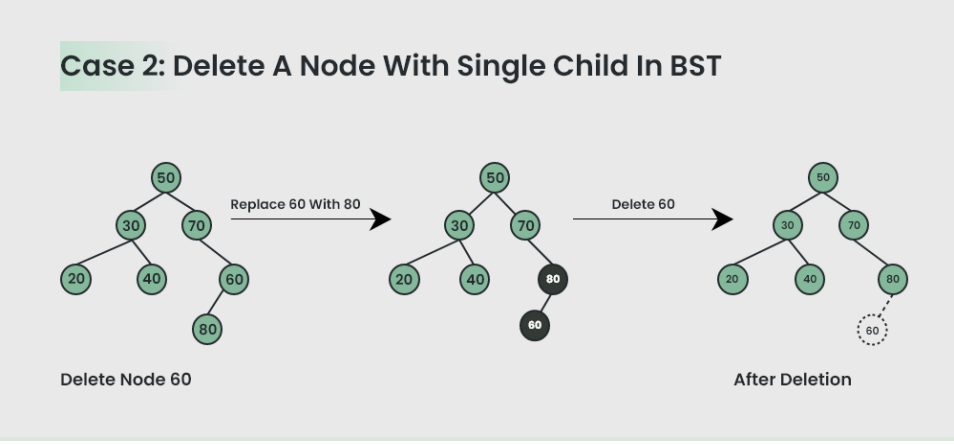
Given a BST, the task is to delete a node in this BST, which can be broken down into 3 scenarios:

Case 1. Delete a Leaf Node in BST



Case 2. Delete a Node with Single Child in BST

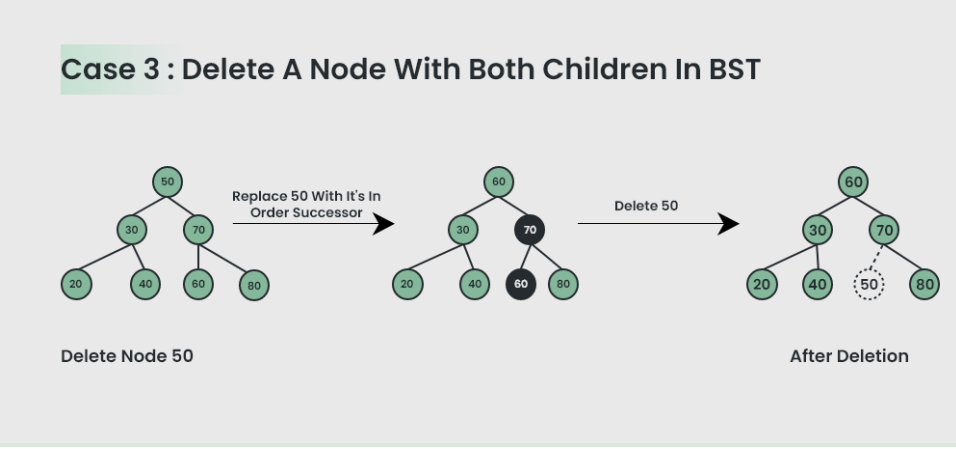
Deleting a single child node is also simple in BST. Copy the child to the node and delete the node.



Case 3. Delete a Node with Both Children in BST

Deleting a node with both children is not so simple. Here we have to delete the node is such a way, that the resulting tree follows the properties of a BST.

The trick is to find the inorder successor of the node. Copy contents of the inorder successor to the node, and delete the inorder successor.



H. Resources/Equipment Required

Sr.No.	Instrument/Equipment /Components/Trainer kit	Specification	Quanti ty
1	*Computer system with operating system*		*1*
2.	*Python IDLE*		*1*

I. Safety and necessary Precautions followed

1. *Turn off power switch only after computer is shut down.*
2. *Do not plug out any computer cables*

J. Source code :

K. Input-Output:

```
Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: a
Enter the number of elements: 5
Enter the elements:
7
4
10
2
6
BST created successfully!

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: b
Enter the element to insert: 8
```

```
Element inserted successfully!

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: c
Pre-order traversal:
7 4 2 6 10 8

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: d
In-order traversal:
2 4 6 7 8 10

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: e
Post-order traversal:
2 6 4 8 10 7

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
```

```
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: f
Enter the element to delete: 4
Element deleted successfully!

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: d
In-order traversal:
2 6 7 8 10

Binary Search Tree Operations:
a. Create a BST
b. Insert an element in BST
c. Pre-order traversal of BST
d. In-order traversal of BST
e. Post-order traversal of BST
f. Delete an element from BST
q. Quit the program

Enter your choice: q
Exiting the program.
```

L. Practical related Quiz.

1. Explain the concept of the "binary search property" in a binary search tree.
2. What are the three common methods to traverse a binary search tree?
3. How do you define the height of a binary search tree?

M. References / Suggestions (lab manual designer should give)

Data structures and algorithms in python, Goodrich, Wiley publication
https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/

N. Assessment-Rubrics

Agenda	Rubric Parameters	Level of Achievement				
		Excellent (5)	**Very Good (4)**	**Good (3)**	**Average (2)**	**Poor (1)**
Problem Understanding	Understand the problem, identify the concept for solution, explain concept and able to identify application of concept.	Able to define & explain concept properly and able to identify application of concept.	Able to identify all the problem/Task and able to apply all concept in problem/Task properly with 6 +very few help.	Able to identify almost all problem/Task and able to apply almost all concept in problem/Task with few exceptions.	Able to identify some problem/Task and able to apply some concept in problem/Task	Able to identify very few problem/ Task and able to apply very few concept in problem/ Task.
Design Methodology	Conceptual design, Division of problem into modules, Selection of design framework	Properly Followed & Properly Justified	Properly Followed & Justified partly	Properly followed & Not Justified	Partially Followed and Partially Justified	Partially followed and Not justified
Implementation	Design, Algorithm, Coding	Properly Followed & Properly implemented	Properly Followed & implemented partly	Properly followed & Not implemented	Partially Followed and Partially implemented	Partially followed and Not implemented
Demonstration	Execution of source code, Working and results	Properly demonstrated & Properly Justified output	Properly demonstrated & Partially Justified output	Partially demonstrated & Justified	Partially demonstrated and Partially Justified	Partially demonstrated and no justification
Viva	Handling Questions	Answered all questions with proper justification	Answered 80% questions	Answered 60% questions	Answered 40% questions	Answered 20% questions

Sign with Date