Subject Code: LPCIT-109

Subject Name: Programming in Java Laboratory

Programme: B. Tech	**L: 0 T: 0 P: 4**
Semester: 5	**Teaching Hours: 48**
Theory/Practical: Practical	**Credits: 2**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 Hours**
Total Marks: 50	**Course Type: Professional Core Course**

Detailed Contents:
Programs to demonstrate:

1. Handling various data types
2. Type casting
3. Arrays – 1D and 2 D
4. Various control structures
5. Various decision structures
6. Recursion
7. Method Overloading by passing objects as arguments
8. Constructor Overloading by passing objects as arguments
9. Various access control and usage of static, final and finalize ( )
10. Command line arguments
11. Various types of inheritance by applying various access controls to its data members and methods
12. Method overriding 1
3. Abstract class
14. Nested class
15. Constructor chaining
16. Importing classes from user defined package and creating packages using access protection
17. Interfaces, nested interfaces and use of extending interfaces
18. Exception Handling - using predefined exception
19. Exception Handling - creating user defined exceptions
20. Multithreading by extending Thread Class
21. Multithreading by implementing Runnable Interface
22. Thread life cycle
23. Applet life cycle
24. Applet for configuring Applets by passing parameters
25. Event Handling
26. Reading and writing from a particular file
27. Database connectivity for various DDL and DML operations
28. String class and its methods
29. StringBuffer class and its methods

Experiment No. 1:

Handling Various Data Types

Code:

```java
public class DataTypesExample {
    public static void main(String[] args) {
        // Integer data type
        int intVar = 100;
        // Floating point data type
        float floatVar = 10.5f;
        // Double data type
        double doubleVar = 20.99;
        // Character data type
        char charVar = 'A';
        // Boolean data type
        boolean boolVar = true;
        // String data type
        String stringVar = "Hello, Java!";

        // Display values
        System.out.println("Integer: " + intVar);
        System.out.println("Float: " + floatVar);
        System.out.println("Double: " + doubleVar);
        System.out.println("Character: " + charVar);
        System.out.println("Boolean: " + boolVar);
        System.out.println("String: " + stringVar);
    }
}
```

Output:

```
Integer: 100

Float: 10.5

Double: 20.99

Character: A

Boolean: true

String: Hello, Java!
```

Experiment No. 2:

Type Casting

Code:

```
public class TypeCastingExample {
    public static void main(String[] args) {
        // Widening (Implicit) Type Casting
        int intVar = 50;
        double doubleVar = intVar; // Automatically converts int
to double

        // Narrowing (Explicit) Type Casting
        double doubleValue = 20.99;
        int intValue = (int) doubleValue; // Explicitly converts
double to int

        // Char to Int Conversion
        char charVar = 'A';
        int charToInt = (int) charVar; // ASCII value of 'A'

        // Int to Char Conversion
        int num = 66;
        char intToChar = (char) num; // Converts ASCII value to
char

        // Display results
        System.out.println("Widening (int to double): " +
doubleVar);
        System.out.println("Narrowing (double to int): " +
intValue);
        System.out.println("Char to Int ('A' to ASCII): " +
charToInt);
        System.out.println("Int to Char (66 to character): " +
intToChar);
```

```
    }
}
```

Output:

```
Widening (int to double): 50.0
Narrowing (double to int): 20
Char to Int ('A' to ASCII): 65
Int to Char (66 to character): B
```

Experiment No. 3:

Arrays – 1D and 2D

Code:

```
public class ArraysExample {
    public static void main(String[] args) {
        // 1D Array
        int[] oneDArray = {10, 20, 30, 40, 50};
        System.out.println("1D Array Elements:");
        for (int num : oneDArray) {
            System.out.print(num + " ");
        }
        System.out.println("\n");

        // 2D Array
        int[][] twoDArray = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        System.out.println("2D Array Elements:");
        for (int i = 0; i < twoDArray.length; i++) {
            for (int j = 0; j < twoDArray[i].length; j++) {
                System.out.print(twoDArray[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Output:

```
1D Array Elements:
10 20 30 40 50

2D Array Elements:
1 2 3
4 5 6
7 8 9
```

Experiment No. 4:

Various Control Structures

Code:

```
public class ControlStructuresExample {
    public static void main(String[] args) {
        // If-Else Statement
        int num = 10;
        if (num > 0) {
            System.out.println("Number is positive");
        } else {
            System.out.println("Number is negative");
        }

        // Switch Case
        int day = 3;
        switch (day) {
            case 1: System.out.println("Monday"); break;
            case 2: System.out.println("Tuesday"); break;
            case 3: System.out.println("Wednesday"); break;
            default: System.out.println("Invalid day");
        }

        // For Loop
        System.out.println("For Loop:");
        for (int i = 1; i <= 5; i++) {
            System.out.print(i + " ");
        }
        System.out.println();

        // While Loop
        System.out.println("While Loop:");
        int count = 1;
        while (count <= 3) {
            System.out.print(count + " ");
            count++;
        }
        System.out.println();

        // Do-While Loop
        System.out.println("Do-While Loop:");
        int value = 1;
        do {
            System.out.print(value + " ");
            value++;
        } while (value <= 3);
        System.out.println();
    }
}
```

Output:

```
Number is positive
Wednesday
For Loop:
1 2 3 4 5
While Loop:
1 2 3
Do-While Loop:
1 2 3
```

Experiment No. 5:

Various Decision Structures

Code:

```java
public class DecisionStructuresExample {
    public static void main(String[] args) {
        int num = 10;

        // If-Else Decision Structure
        if (num > 0) {
            System.out.println("The number is positive.");
        } else if (num < 0) {
            System.out.println("The number is negative.");
        } else {
            System.out.println("The number is zero.");
        }

        // Nested If-Else
        int age = 20;
        if (age >= 18) {
            if (age >= 60) {
                System.out.println("Senior Citizen");
            } else {
                System.out.println("Adult");
            }
        } else {
            System.out.println("Minor");
        }

        // Switch Case Decision Structure
        int grade = 'B';
        switch (grade) {
            case 'A': System.out.println("Excellent!"); break;
            case 'B': System.out.println("Good Job!"); break;
            case 'C': System.out.println("Fair"); break;
            case 'D': System.out.println("Needs Improvement");
break;
            default: System.out.println("Invalid Grade");
        }
    }
}
```

Output:

```
The number is positive.
Adult
Good Job!
```

Experiment No. 6:

Recursion

Code:

```
public class RecursionExample {
    // Recursive method to calculate factorial
    public static int factorial(int n) {
        if (n == 0 || n == 1) {
            return 1; // Base case
        }
        return n * factorial(n - 1); // Recursive call
    }

    // Recursive method to calculate Fibonacci series
    public static int fibonacci(int n) {
        if (n == 0) {
            return 0; // Base case
        } else if (n == 1) {
            return 1; // Base case
        }
        return fibonacci(n - 1) + fibonacci(n - 2); // Recursive
call
    }

    public static void main(String[] args) {
        int num = 5;

        // Factorial calculation
        System.out.println("Factorial of " + num + " is: " +
factorial(num));

        // Fibonacci series up to n terms
        System.out.print("Fibonacci series up to " + num + " 
terms: ");

        for (int i = 0; i < num; i++) {

            System.out.print(fibonacci(i) + " ");

        }

    }

}
```

Output:

```
Factorial of 5 is: 120

Fibonacci series up to 5 terms: 0 1 1 2 3
```

```
}
```

Experiment No. 7:

Method Overloading by Passing Objects as Arguments

Code:

```
class Box {
    double width, height, depth;

    // Constructor to initialize dimensions
    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    // Method to calculate volume
    double volume() {
        return width * height * depth;
    }

    // Method Overloading: Passing Object as Argument
    boolean isSameVolume(Box b) {
        return this.volume() == b.volume();
    }

    // Overloaded method with different argument
    boolean isLarger(Box b) {
        return this.volume() > b.volume();
    }
}

public class MethodOverloadingExample {
    public static void main(String[] args) {
        Box box1 = new Box(3, 4, 5);
        Box box2 = new Box(3, 4, 5);
        Box box3 = new Box(5, 6, 7);


        System.out.println("Box1 and Box2 have the same volume: "
+ box1.isSameVolume(box2));
        System.out.println("Box1 is larger than Box3: " +
box1.isLarger(box3));
    }
}
```

Output:

```
Box1 and Box2 have the same volume: true
Box1 is larger than Box3: false
```

Experiment No. 8:

Constructor Overloading by Passing Objects as Arguments

Code:

```
class Box {
    double width, height, depth;

    // Constructor 1: Default Constructor
    Box() {
        width = height = depth = 0;
    }

    // Constructor 2: Parameterized Constructor
    Box(double w, double h, double d) {
        width = w;
        height = h;
        depth = d;
    }

    // Constructor 3: Copy Constructor (Passing Object as
Argument)
    Box(Box b) {
        this.width = b.width;
        this.height = b.height;
        this.depth = b.depth;
    }

    // Method to calculate volume
    double volume() {
        return width * height * depth;
    }
}
```

```
public class ConstructorOverloadingExample {
    public static void main(String[] args) {
        Box box1 = new Box(3, 4, 5); // Using parameterized
constructor
        Box box2 = new Box(box1);    // Using copy constructor


        System.out.println("Volume of Box1: " + box1.volume());
        System.out.println("Volume of Box2 (copied from Box1): " +
box2.volume());
    }
}
```

Output:

```
Volume of Box1: 60.0
Volume of Box2 (copied from Box1): 60.0
```

Experiment No. 9:

Various Access Control and Usage of Static, Final, and finalize()

Code:

```
class Example {
    // Private variable (Access Control)
    private int privateVar = 10;

    // Public variable
    public int publicVar = 20;

    // Protected variable
    protected int protectedVar = 30;

    // Static variable
    static int staticVar = 50;

    // Final variable (constant)
    final int finalVar = 100;

    // Constructor
    Example() {
        System.out.println("Constructor called!");
    }

    // Method to demonstrate access control
    void display() {
        System.out.println("Private Variable: " + privateVar);
        System.out.println("Public Variable: " + publicVar);
        System.out.println("Protected Variable: " + protectedVar);
        System.out.println("Static Variable: " + staticVar);
        System.out.println("Final Variable: " + finalVar);
    }

    // finalize() method
    protected void finalize() {
        System.out.println("finalize() method called before object
is garbage collected.");
    }
}
public class AccessControlExample {
    public static void main(String[] args) {
        Example obj = new Example();
        obj.display();

        // Accessing static variable using class name
        System.out.println("Accessing Static Variable: " +
Example.staticVar);

        // Demonstrating finalize()
        obj = null;  // Making object eligible for garbage
collection
        System.gc(); // Requesting garbage collection
    }
}
```

Output:

```
Constructor called!
Private Variable: 10
Public Variable: 20
Protected Variable: 30
Static Variable: 50
Final Variable: 100
Accessing Static Variable: 50
finalize() method called before object is garbage collected.
```

Experiment No. 10:

Command Line Arguments

Code:

```
public class CommandLineArguments {
    public static void main(String[] args) {
        // Check if arguments are provided
        if (args.length == 0) {
            System.out.println("No command line arguments provided.");
            return;
        }

        // Display command line arguments
        System.out.println("Command Line Arguments:");
        for (int i = 0; i < args.length; i++) {
            System.out.println("Argument " + (i + 1) + ": " + args[i]);
        }
    }
}
```

Execution Command (Example in Terminal/Command Prompt):

```
java CommandLineArguments Hello World 123
```

Output:

```
Command Line Arguments:
Argument 1: Hello
Argument 2: World
Argument 3: 123
```

Experiment No. 11:

Various Types of Inheritance by Applying Various Access Controls to Data Members and Methods

Code:

```
// Parent class demonstrating different access control levels
class Parent {
    private int privateVar = 10;  // Private variable (not inherited)
    protected int protectedVar = 20;  // Protected variable (inherited)
    public int publicVar = 30;  // Public variable (inherited)

    // Private method (not inherited)
    private void privateMethod() {
        System.out.println("Private Method of Parent");
    }

    // Protected method (can be inherited)
    protected void protectedMethod() {
        System.out.println("Protected Method of Parent");
    }

    // Public method (can be inherited)
    public void publicMethod() {
        System.out.println("Public Method of Parent");
    }
}

// Single Inheritance
class Child extends Parent {
    public void display() {
        // System.out.println("Private Variable: " +
privateVar); // Not accessible
        System.out.println("Protected Variable: " + protectedVar);
        System.out.println("Public Variable: " + publicVar);

        // privateMethod(); // Not accessible
        protectedMethod();  // Accessible
        publicMethod();  // Accessible
    }
}

// Multilevel Inheritance
class GrandChild extends Child {
    public void show() {
        System.out.println("Accessing inherited members in
GrandChild:");
        System.out.println("Protected Variable: " + protectedVar);
        System.out.println("Public Variable: " + publicVar);
    }
}

// Hierarchical Inheritance
class AnotherChild extends Parent {
    public void anotherDisplay() {
        System.out.println("Accessing from AnotherChild:");
        System.out.println("Protected Variable: " + protectedVar);
        System.out.println("Public Variable: " + publicVar);
    }
}

public class InheritanceExample {
    public static void main(String[] args) {
        // Single Inheritance
        Child child = new Child();
        System.out.println("Single Inheritance:");
        child.display();

        // Multilevel Inheritance
        GrandChild grandChild = new GrandChild();
        System.out.println("\nMultilevel Inheritance:");
        grandChild.show();

        // Hierarchical Inheritance
        AnotherChild anotherChild = new AnotherChild();
        System.out.println("\nHierarchical Inheritance:");
        anotherChild.anotherDisplay();
    }
}
```

Output:

```
Single Inheritance:
Protected Variable: 20
Public Variable: 30
Protected Method of Parent
Public Method of Parent

Multilevel Inheritance:
Accessing inherited members in GrandChild:
Protected Variable: 20
Public Variable: 30

Hierarchical Inheritance:
Accessing from AnotherChild:
Protected Variable: 20
Public Variable: 30
```

Experiment No. 12:

Method Overriding

Code:

```
// Parent class
class Animal {
    // Method to be overridden
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

// Child class overriding the makeSound() method
class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Dog barks");
    }
}

// Another child class overriding the makeSound() method
class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Cat meows");
    }
}

public class MethodOverridingExample {
    public static void main(String[] args) {
        // Parent class reference, child class object
        Animal myAnimal = new Animal();
        Animal myDog = new Dog();
        Animal myCat = new Cat();

        System.out.println("Method Overriding Example:");
        myAnimal.makeSound(); // Calls parent class method
        myDog.makeSound();    // Calls Dog's overridden method
        myCat.makeSound();    // Calls Cat's overridden method
    }
}
```

Output:

```
Method Overriding Example:
Animal makes a sound
Dog barks
Cat meows
```

Experiment No. 13:

Abstract Class

Code:

```
// Abstract class
abstract class Shape {
    // Abstract method (must be implemented by subclasses)
    abstract void draw();

    // Concrete method (can be used directly)
    public void display() {
        System.out.println("This is a shape.");
    }
}

// Subclass 1: Circle
class Circle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a Circle");
    }
}

// Subclass 2: Rectangle
class Rectangle extends Shape {
    @Override
    void draw() {
        System.out.println("Drawing a Rectangle");
    }
}

public class AbstractClassExample {
    public static void main(String[] args) {
        // Cannot instantiate abstract class
        // Shape shape = new Shape();  // This will cause an error

        // Using subclasses
        Shape circle = new Circle();
        Shape rectangle = new Rectangle();

        System.out.println("Abstract Class Example:");
        circle.display();
        circle.draw();

        rectangle.display();
        rectangle.draw();
    }
}
```

Output:

```
Abstract Class Example:
This is a shape.
Drawing a Circle
This is a shape.
Drawing a Rectangle
```

Experiment No. 14:

Nested Class

Code:

```
// Outer class
class OuterClass {
    private String outerMessage = "Hello from Outer Class";

    // Inner class
    class InnerClass {
        void display() {
            System.out.println("Accessing Outer Class Variable: "
+ outerMessage);
        }
    }

    // Static nested class
    static class StaticNestedClass {
        void show() {
            System.out.println("Inside Static Nested Class");
        }
    }
}

public class NestedClassExample {
    public static void main(String[] args) {
        // Creating an instance of the Inner Class
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner = outer.new InnerClass();
        System.out.println("Inner Class:");
        inner.display();
        // Creating an instance of the Static Nested Class
        OuterClass.StaticNestedClass staticNested = new
OuterClass.StaticNestedClass();
        System.out.println("\nStatic Nested Class:");
        staticNested.show();
    }
}
```

Output:

```
Inner Class:
Accessing Outer Class Variable: Hello from Outer Class

Static Nested Class:
Inside Static Nested Class
```

Experiment No. 15:

Constructor Chaining

Code:

```
class Parent {
    // Parent class constructor
    Parent() {
        System.out.println("Parent class constructor called");
    }
}

class Child extends Parent {
    // Child class constructor with no arguments
    Child() {
        this(10); // Calling parameterized constructor of the same
class
        System.out.println("Child class default constructor
called");
    }

    // Parameterized constructor
    Child(int x) {
        super(); // Calling Parent class constructor
        System.out.println("Child class parameterized constructor
called with value: " + x);
    }
}

public class ConstructorChainingExample {
    public static void main(String[] args) {
        System.out.println("Constructor Chaining Example:");
        Child obj = new Child();
    }
}
```

Output:

```
Constructor Chaining Example:

Parent class constructor called

Child class parameterized constructor called with value: 10

Child class default constructor called
```

Experiment No. 16:

Importing Classes from User-Defined Package and Creating Packages Using Access Protection

Step 1: Create a Package

Create a directory named `mypackage`, and inside it, create a Java file `MyClass.java`.

Code for MyClass.java (Inside mypackage)

```
// Define a package
package mypackage;


// Public class inside the package
public class MyClass {
    // Protected method (can be accessed within package and 
subclasses)
    protected void protectedMethod() {
        System.out.println("This is a protected method from 
MyClass in mypackage.");
    }


    // Public method (can be accessed anywhere)
    public void publicMethod() {
        System.out.println("This is a public method from MyClass 
in mypackage.");
    }
}
```

Step 2: Use the Package in Another Java File

Create another Java file outside `mypackage` (in the main directory) to import and use the class.

Code for MainClass.java

```
// Import the package
import mypackage.MyClass;


public class MainClass extends MyClass {
    public static void main(String[] args) {
        System.out.println("Importing Classes from User-Defined
Package Example:");

        // Creating an object of MyClass
        MyClass obj = new MyClass();
        obj.publicMethod();  // Public method is accessible

        // Creating an object of MainClass to access protected
method
        MainClass mainObj = new MainClass();
        mainObj.protectedMethod(); // Protected method can be
accessed through inheritance
    }
}
```

How to Compile and Run

1. **Compile the package class:**

   ```
   javac -d . MyClass.java
   ```

2. **Compile the main class:**

   ```
   javac MainClass.java
   ```

3. **Run the program:**

   ```
   java MainClass
   ```

Output:

```
Importing Classes from User-Defined Package Example:
This is a public method from MyClass in mypackage.
This is a protected method from MyClass in mypackage.
```

Experiment No. 17:

Interfaces, Nested Interfaces, and Use of Extending Interfaces

Code:

```
// Defining an interface
interface Animal {
    void makeSound(); // Abstract method
}

// Interface extending another interface
interface Pet extends Animal {
    void play();
}

// Implementing the Pet interface
class Dog implements Pet {
    @Override
    public void makeSound() {
        System.out.println("Dog barks");
    }

    @Override
    public void play() {
        System.out.println("Dog is playing");
    }
}

// Outer interface containing a nested interface
interface OuterInterface {
    void outerMethod();

    // Nested Interface
    interface NestedInterface {
        void nestedMethod();
    }
}

// Class implementing the Nested Interface
class NestedClass implements OuterInterface.NestedInterface {
    @Override
    public void nestedMethod() {
        System.out.println("Nested Interface Method
Implementation");
    }
}

public class InterfaceExample {
    public static void main(String[] args) {
        System.out.println("Interfaces, Nested Interfaces, and
Extending Interfaces Example:");

        // Using a class that implements an interface
        Dog myDog = new Dog();
        myDog.makeSound();
        myDog.play();

        // Using a class that implements a nested interface
        NestedClass nestedObj = new NestedClass();
        nestedObj.nestedMethod();
    }
}
```

Output:

```
Interfaces, Nested Interfaces, and Extending Interfaces Example:

Dog barks

Dog is playing

Nested Interface Method Implementation
```

Experiment No.18:

Exception Handling Using Predefined Exception

Code:

```
public class PredefinedExceptionExample {

    public static void main(String[] args) {

        System.out.println("Exception Handling Using Predefined 
Exception Example:");


        try {

            // Division by zero - will cause ArithmeticException

            int result = 10 / 0;

            System.out.println("Result: " + result);

        } catch (ArithmeticException e) {

            System.out.println("Exception Caught: " + e);

        }


        try {

            // Accessing an invalid index in an array - will cause 
ArrayIndexOutOfBoundsException

            int[] numbers = {1, 2, 3};

            System.out.println(numbers[5]); // Invalid index

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Exception Caught: " + e);

        }


        try {

            // Null reference access - will cause 
NullPointerException

            String str = null;

            System.out.println(str.length()); // Trying to get 
length of null

        } catch (NullPointerException e) {
            System.out.println("Exception Caught: " + e);
        }


        System.out.println("Program execution continues normally 
after exception handling.");
    }
}
```

Output:

```
Exception Handling Using Predefined Exception Example:

Exception Caught: java.lang.ArithmeticException: / by zero

Exception Caught: java.lang.ArrayIndexOutOfBoundsException: Index 
5 out of bounds for length 3

Exception Caught: java.lang.NullPointerException

Program execution continues normally after exception handling.
```

Experiment No. 19:

Exception Handling - Creating User-Defined Exceptions

Code:

```
// Custom Exception Class
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

public class UserDefinedExceptionExample {
    // Method to check age eligibility
    static void checkAge(int age) throws InvalidAgeException {
        if (age < 18) {
            throw new InvalidAgeException("Age is less than 18.
Not eligible to vote.");
        } else {
            System.out.println("Eligible to vote.");
        }
    }

    public static void main(String[] args) {
        System.out.println("Exception Handling Using User-Defined
Exception Example:");

        try {
            checkAge(16);  // This will throw an exception
        } catch (InvalidAgeException e) {
            System.out.println("Exception Caught: " +
e.getMessage());
        }
        try {
            checkAge(20);  // This will not throw an exception
        } catch (InvalidAgeException e) {
            System.out.println("Exception Caught: " +
e.getMessage());
        }
    }
}
```

Output:

```
Exception Handling Using User-Defined Exception Example:
Exception Caught: Age is less than 18. Not eligible to vote.
Eligible to vote.
```

Experiment No. 20:

Multithreading by Extending Thread Class

Code:

```
// Creating a thread by extending the Thread class
class MyThread extends Thread {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() +
" - Count: " + i);
            try {
                Thread.sleep(1000); // Pause for 1 second
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

public class ThreadExample {
    public static void main(String[] args) {
        System.out.println("Multithreading by Extending Thread
Class Example:");

        // Creating and starting two threads
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();

        t1.setName("Thread 1");
        t2.setName("Thread 2");

        t1.start();
        t2.start();
    }
}
```

Output:

```
Multithreading by Extending Thread Class Example:
Thread 1 - Count: 1
Thread 2 - Count: 1
Thread 1 - Count: 2
Thread 2 - Count: 2
Thread 1 - Count: 3
Thread 2 - Count: 3
Thread 1 - Count: 4
Thread 2 - Count: 4
Thread 1 - Count: 5
Thread 2 - Count: 5
```

Experiment No. 21: \

Multithreading by Implementing Runnable Interface

Code:

```
// Creating a thread by implementing the Runnable interface
class MyRunnable implements Runnable {
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() +
" - Count: " + i);
            try {
                Thread.sleep(1000); // Pause for 1 second
            } catch (InterruptedException e) {
                System.out.println(e);
            }
        }
    }
}

public class RunnableExample {
    public static void main(String[] args) {
        System.out.println("Multithreading by Implementing
Runnable Interface Example:");

        // Creating Runnable objects
        MyRunnable runnable1 = new MyRunnable();
        MyRunnable runnable2 = new MyRunnable();

        // Creating Thread objects and passing Runnable instances
        Thread t1 = new Thread(runnable1, "Thread 1");
        Thread t2 = new Thread(runnable2, "Thread 2");

        // Starting the threads
        t1.start();
        t2.start();
    }
}
```

Output:

```
Multithreading by Implementing Runnable Interface Example:
Thread 1 - Count: 1
Thread 2 - Count: 1
Thread 1 - Count: 2
Thread 2 - Count: 2
Thread 1 - Count: 3
Thread 2 - Count: 3
Thread 1 - Count: 4
Thread 2 - Count: 4
Thread 1 - Count: 5
Thread 2 - Count: 5
```

Experiment No. 22:

Thread Life Cycle

Code:

```
class ThreadLifecycle extends Thread {
    public void run() {
        System.out.println(Thread.currentThread().getName() + " is
in RUNNING state.");
        try {
            Thread.sleep(2000); // Thread goes to TIMED WAITING
state
            System.out.println(Thread.currentThread().getName() +
" woke up from sleep.");
        } catch (InterruptedException e) {
            System.out.println(e);
        }
        System.out.println(Thread.currentThread().getName() + " is
TERMINATED.");
    }
}

public class ThreadLifeCycleExample {
    public static void main(String[] args) {
        System.out.println("Thread Life Cycle Example:");

        // Thread is in NEW state
        ThreadLifecycle thread = new ThreadLifecycle();
        System.out.println("Thread state after creation: " +
thread.getState());

        // Thread is in RUNNABLE state
        thread.start();
        System.out.println("Thread state after start(): " +
thread.getState());
```

```
        // Checking thread state during execution
        try {
            Thread.sleep(500);
            System.out.println("Thread state while running: " +
thread.getState());

            thread.join(); // Waiting for the thread to complete
        } catch (InterruptedException e) {
            System.out.println(e);
        }

        // Thread is in TERMINATED state after execution
        System.out.println("Thread state after termination: " +
thread.getState());
    }
}
```

Output:

```
Thread Life Cycle Example:
Thread state after creation: NEW
Thread state after start(): RUNNABLE
Thread-0 is in RUNNING state.
Thread state while running: RUNNABLE
Thread-0 woke up from sleep.
Thread-0 is TERMINATED.
Thread state after termination: TERMINATED
```

Experiment No. 23:

Applet Life Cycle

Code:

```
import java.applet.Applet;
import java.awt.Graphics;

public class AppletLifeCycle extends Applet {
    // Called when the applet is initialized
    public void init() {
        System.out.println("Applet is initialized.");
    }

    // Called when the applet starts
    public void start() {
        System.out.println("Applet has started.");
    }

    // Called when the applet needs to be repainted
    public void paint(Graphics g) {
        System.out.println("Applet is being painted.");
        g.drawString("Applet Life Cycle Example", 50, 50);
    }

    // Called when the applet is stopped
    public void stop() {
        System.out.println("Applet has stopped.");
    }

    // Called when the applet is destroyed
    public void destroy() {
        System.out.println("Applet is destroyed.");
    }
}
```

How to Run the Applet:

1. **Compile the applet:**

   ```
   javac AppletLifeCycle.java
   ```

2. **Create an HTML file (`AppletExample.html`) to load the applet:**

   ```
   <html>
   <body>
       <applet code="AppletLifeCycle.class" width="300"
   height="200"></applet>
   </body>
   </html>
   ```

3. **Run the applet using the applet viewer:**

   ```
   appletviewer AppletExample.html
   ```

Output (Console Log):

```
Applet is initialized.
Applet has started.
Applet is being painted.
Applet has stopped.
Applet is destroyed.
```

Experiment No. 24:

Applet for Configuring Applets by Passing Parameters

Code:

```
import java.applet.Applet;

import java.awt.Graphics;


public class ParameterizedApplet extends Applet {

    String message;


    // Called to initialize the applet

    public void init() {

        // Getting parameter from HTML file

        message = getParameter("message");

        if (message == null) {

            message = "Default Message: No Parameter Passed!";

        }

    }


    // Called to paint content on the applet window

    public void paint(Graphics g) {

        g.drawString("Passed Parameter: " + message, 50, 50);

    }

}
```

Creating an HTML File (`AppletParameter.html`) to Pass Parameters:

```
<html>

<body>

    <applet code="ParameterizedApplet.class" width="400"
height="200">

        <param name="message" value="Hello, Welcome to Applet
Programming!">

    </applet>
```

```
</body>
</html>
```

How to Run the Applet:

1. **Compile the Java file:**
   ```
   javac ParameterizedApplet.java
   ```
2. **Run the applet using appletviewer:**
   ```
   appletviewer AppletParameter.html
   ```

Output:

The applet window will display:

```
Passed Parameter: Hello, Welcome to Applet Programming!
Passed Parameter: Default Message: No Parameter Passed!
```

Experiment No. 25:

Event Handling

Code:

```
import java.applet.Applet;

import java.awt.*;

import java.awt.event.*;


public class EventHandlingApplet extends Applet implements
ActionListener {

    Button btnClick;

    Label lblMessage;


    // Initialize the applet and add components

    public void init() {

        setLayout(new FlowLayout());


        btnClick = new Button("Click Me");

        lblMessage = new Label("Click the button to see the
event!");


        add(btnClick);

        add(lblMessage);


        // Registering event listener

        btnClick.addActionListener(this);

    }


    // Handling button click event

    public void actionPerformed(ActionEvent e) {

        lblMessage.setText("Button Clicked! Event Handled.");

    }

}
```

Creating an HTML File (`EventHandling.html`) to Run the Applet:

```
<html>

<body>

    <applet code="EventHandlingApplet.class" width="400"
height="200"></applet>

</body>

</html>
```

How to Run the Applet:

1. **Compile the Java file:**

   ```
   javac EventHandlingApplet.java
   ```

2. **Run the applet using appletviewer:**

   ```
   appletviewer EventHandling.html
   ```

Output:

- Initially, the applet displays a **"Click Me"** button and a label:

  ```
  Click the button to see the event!
  ```

- When the button is clicked, the label changes to:

  ```
  Button Clicked! Event Handled.
  ```

Experiment No. 26:

Reading and Writing from a Particular File

Code:

```
import java.io.*;


public class FileReadWrite {
    public static void main(String[] args) {
        String fileName = "example.txt";


        // Writing to a file
        try (FileWriter writer = new FileWriter(fileName)) {
            writer.write("Hello, this is a file handling 
experiment in Java.\n");
            writer.write("Writing to a file successfully!\n");
            System.out.println("Data successfully written to " + 
fileName);
        } catch (IOException e) {
            System.out.println("Error writing to file: " + 
e.getMessage());
        }


        // Reading from a file
        try (BufferedReader reader = new BufferedReader(new 
FileReader(fileName))) {
            String line;
            System.out.println("\nReading from " + fileName + 
":");
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
        } catch (IOException e) {
            System.out.println("Error reading from file: " + 
e.getMessage());
```

```
        }
    }
}
```

Output:

```
Data successfully written to example.txt


Reading from example.txt:
Hello, this is a file handling experiment in Java.
Writing to a file successfully!
```

Experiment No. 27:

Database Connectivity for Various DDL and DML Operations

Code:

```
import java.sql.*;


public class DatabaseOperations {
    public static void main(String[] args) {
        // Database connection details
        String url = "jdbc:mysql://localhost:3306/testdb"; //
Change to your database
        String user = "root"; // Change to your MySQL username
        String password = ""; // Change to your MySQL password


        try {
            // Load MySQL JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");


            // Establish Connection
            Connection conn = DriverManager.getConnection(url,
user, password);
            Statement stmt = conn.createStatement();


            // DDL Operation: Creating a Table
            String createTableSQL = "CREATE TABLE IF NOT EXISTS
Students (" +
                                    "ID INT PRIMARY KEY
AUTO_INCREMENT, " +
                                    "Name VARCHAR(50), " +
                                    "Age INT)";
            stmt.executeUpdate(createTableSQL);
            System.out.println("Table 'Students' created
successfully.");
```

```
            // DML Operation: Inserting Data
            String insertSQL = "INSERT INTO Students (Name, Age) 
VALUES ('John Doe', 22), ('Alice Smith', 20)";
            int rowsInserted = stmt.executeUpdate(insertSQL);
            System.out.println(rowsInserted + " rows inserted.");

            // DML Operation: Retrieving Data
            String selectSQL = "SELECT * FROM Students";
            ResultSet rs = stmt.executeQuery(selectSQL);
            System.out.println("\nStudent Records:");
            while (rs.next()) {
                System.out.println("ID: " + rs.getInt("ID") + ", 
Name: " + rs.getString("Name") + ", Age: " + rs.getInt("Age"));
            }

            // DML Operation: Updating Data
            String updateSQL = "UPDATE Students SET Age = 23 WHERE 
Name = 'John Doe'";
            int rowsUpdated = stmt.executeUpdate(updateSQL);
            System.out.println("\n" + rowsUpdated + " row(s) 
updated.");

            // DML Operation: Deleting Data
            String deleteSQL = "DELETE FROM Students WHERE Name = 
'Alice Smith'";
            int rowsDeleted = stmt.executeUpdate(deleteSQL);
            System.out.println(rowsDeleted + " row(s) deleted.");

            // Closing connection
            rs.close();
            stmt.close();
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Table 'Students' created successfully.
2 rows inserted.

Student Records:
ID: 1, Name: John Doe, Age: 22
ID: 2, Name: Alice Smith, Age: 20

1 row(s) updated.
1 row(s) deleted.
```

Experiment No.28:

String Class and Its Methods

Code:

```
public class StringMethodsExample {
    public static void main(String[] args) {
        // Creating strings
        String str1 = "Hello, World!";
        String str2 = " Java Programming ";

        // String length
        System.out.println("Length of str1: " + str1.length());

        // Convert to uppercase and lowercase
        System.out.println("Uppercase: " + str1.toUpperCase());
        System.out.println("Lowercase: " + str1.toLowerCase());

        // Substring extraction
        System.out.println("Substring (7,12): " +
str1.substring(7, 12));

        // Checking if string contains a substring
        System.out.println("Contains 'World': " +
str1.contains("World"));

        // Replacing characters
        System.out.println("Replace 'World' with 'Java': " +
str1.replace("World", "Java"));

        // Removing whitespace
        System.out.println("Trimmed str2: '" + str2.trim() + "'");

        // Checking if a string is empty
        String emptyStr = "";
        System.out.println("Is emptyStr empty? " +
emptyStr.isEmpty());

        // Concatenation
        System.out.println("Concatenation: " +
str1.concat(str2.trim()));

        // Character at a specific index
        System.out.println("Character at index 4: " +
str1.charAt(4));

        // Splitting a string
        String[] words = str1.split(" ");
        System.out.println("Words in str1:");
        for (String word : words) {
            System.out.println(word);
        }

        // Comparing strings
        String str3 = "hello, world!";
        System.out.println("Equals (case-sensitive): " +
str1.equals(str3));
        System.out.println("Equals (case-insensitive): " +
str1.equalsIgnoreCase(str3));
    }
}
```

Output:

```
Length of str1: 13

Uppercase: HELLO, WORLD!

Lowercase: hello, world!

Substring (7,12): World

Contains 'World': true

Replace 'World' with 'Java': Hello, Java!

Trimmed str2: 'Java Programming'

Is emptyStr empty? true

Concatenation: Hello, World!Java Programming

Character at index 4: o

Words in str1:

Hello,

World!

Equals (case-sensitive): false

Equals (case-insensitive): true
```

Experiment No. 29:

StringBuffer Class and Its Methods

Code:

```
public class StringBufferMethodsExample {
    public static void main(String[] args) {
        // Creating a StringBuffer
        StringBuffer sb = new StringBuffer("Hello");

        // Append text
        sb.append(" World!");
        System.out.println("After append: " + sb);

        // Insert text at a specific position
        sb.insert(6, "Java ");
        System.out.println("After insert: " + sb);

        // Replace a part of the string
        sb.replace(6, 10, "Amazing");
        System.out.println("After replace: " + sb);

        // Delete a part of the string
        sb.delete(6, 13);
        System.out.println("After delete: " + sb);

        // Reverse the string
        sb.reverse();
        System.out.println("After reverse: " + sb);

        // Restore the original string
        sb.reverse();
        // Get length and capacity
        System.out.println("Length: " + sb.length());
        System.out.println("Capacity: " + sb.capacity());

        // Set length (truncate or extend)
        sb.setLength(5);
        System.out.println("After setLength(5): " + sb);
    }
}
```

Output:

```
After append: Hello World!
After insert: Hello Java World!
After replace: Hello Amazing World!
After delete: Hello World!
After reverse: !dlroW olleH
Length: 12
Capacity: 31
After setLength(5): Hello
```

Subject Code: LPCIT-111

Subject Name: Internet of Things Laboratory

Programme: B. Tech	**L: 0 T: 0 P: 4**
Semester: 5	**Teaching Hours: 48**
Theory/Practical: Practical	**Credits: 2**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 Hours**
Total Marks: 50	**Course Type: Professional Core Course**

Detailed Contents:

1. Familiarization with Arduino/ Raspberry Pi and perform necessary software installation
2. To demonstrate the communication modules like BLE, WIFI, XBEE
3. To find the IP address of Computer/ other devices
4. To interface LED/ Buzzer with Arduino/ Raspberry Pi and write a program to turn ON/OFF LED for specific duration.
5. To interface DHT11/ DHT22 sensor with Arduino/ Raspberry Pi and write a program to print temperature and humidity readings.
6. To interface PIR Sensor with Arduino/ Raspberry Pi and write a program to check the motion of PIR sensor.
7. To interface PI Camera with Arduino/ Raspberry Pi and write a program to start the camera and to place the clicked pictures on the desktop.
8. To transmit and access the sensed data to any cloud platform.

Experiment No. 1:

Familiarization with Arduino/Raspberry Pi and Perform Necessary Software Installation

Arduino Installation Steps:

1. **Download Arduino IDE**
    - Visit Arduino Official Website
    - Download the latest Arduino IDE for your OS (Windows, macOS, or Linux).
2. **Install the Arduino IDE**
    - Run the downloaded installer and follow the on-screen instructions.
    - Install necessary drivers when prompted.
3. **Connect Arduino Board**
    - Use a USB cable to connect the Arduino board (Uno, Mega, etc.) to your computer.
    - Open the Arduino IDE.
4. **Select the Correct Board and Port**
    - Go to **Tools** > **Board** and select your Arduino model (e.g., Arduino Uno).
    - Go to **Tools** > **Port** and select the correct COM port.
5. **Upload a Test Sketch**
    - Open **File** > **Examples** > **Basics** > **Blink**
    - Click the **Upload** button.
    - If the onboard LED blinks, the setup is successful.

Raspberry Pi Setup:

1. **Download and Install Raspberry Pi OS**
    - Download Raspberry Pi Imager
    - Insert a microSD card into your computer and use Raspberry Pi Imager to flash the OS.
2. **Boot Raspberry Pi**
    - Insert the microSD card into the Raspberry Pi and power it on.
    - Connect a monitor, keyboard, and mouse (if using GUI).
3. **Basic Configuration**
    - Open the terminal and update the system:

      ```
      sudo apt update && sudo apt upgrade -y
      ```

- Enable SSH for remote access:

  ```
  sudo raspi-config
  ```

- Navigate to **Interfacing Options** and enable **SSH, I2C, and SPI** if required.

4. **Python and GPIO Library Installation**

   - Install Python and GPIO libraries:

     ```
     sudo apt install python3-pip
     pip3 install Rpi.GPIO
     ```

Experiment No.2:

To Demonstrate the Communication Modules like BLE, Wi-Fi, and XBee

1. Bluetooth Low Energy (BLE) Communication using Arduino

Hardware Required:

- Arduino Uno/Nano
- HC-05 (Bluetooth Module)
- Jumper Wires

Connections:

HC-05 Pin	Arduino Pin
VCC	5V
GND	GND
TX	RX (Pin 10)
RX	TX (Pin 11)

Code:

```
#include <SoftwareSerial.h>
SoftwareSerial BT(10, 11); // RX, TX

void setup() {
    Serial.begin(9600);
    BT.begin(9600);
    Serial.println("Bluetooth Module Ready");
}

void loop() {
    if (BT.available()) {
        char data = BT.read();
        Serial.print("Received: ");
        Serial.println(data);
    }
}
```

Output:

- When data is sent from a Bluetooth terminal app, it appears in the serial monitor.

2. Wi-Fi Communication using ESP8266

Hardware Required:

- ESP8266 Wi-Fi Module
- Arduino Uno

Connections:

ESP8266 Pin	Arduino Pin
VCC	3.3V
GND	GND
TX	RX (Pin 10)
RX	TX (Pin 11)

Code:

```
#include <SoftwareSerial.h>

SoftwareSerial ESP(10, 11); // RX, TX


void setup() {

    Serial.begin(115200);

    ESP.begin(115200);

    Serial.println("Connecting to Wi-Fi...");

    ESP.println("AT+CWMODE=1"); // Set as station mode

    ESP.println("AT+CWJAP=\"Your_SSID\",\"Your_PASSWORD\""); //
Connect to Wi-Fi

}


void loop() {

    if (ESP.available()) {

        Serial.write(ESP.read());

    }

}
```

Output:

- The serial monitor displays the Wi-Fi connection status.

3. XBee Communication

Hardware Required:

- XBee Module (2 units)
- Arduino Uno (2 units)

Connections:

XBee Pin	Arduino Pin
VCC	3.3V
GND	GND
TX	RX (Pin 10)
RX	TX (Pin 11)

Code for Transmitter:

```
#include <SoftwareSerial.h>
SoftwareSerial XBee(10, 11);

void setup() {
    XBee.begin(9600);
}

void loop() {
    XBee.println("Hello from XBee Transmitter");
    delay(1000);
}
```

Code for Receiver:

```
#include <SoftwareSerial.h>
SoftwareSerial XBee(10, 11);

void setup() {
    Serial.begin(9600);
    XBee.begin(9600);
}

void loop() {
    if (XBee.available()) {
        Serial.print("Received: ");
        Serial.println(XBee.readString());
    }
}
```

Output:

- The receiver displays "Hello from XBee Transmitter" in the serial monitor.

Experiment No. 3:

To Find the IP Address of a Computer/Other Devices

1. Finding IP Address on Windows

Using Command Prompt (CMD):

1. Open **Command Prompt** (`Win + R` → type `cmd` → press Enter).
2. Type the following command and press Enter:
   ```
   ipconfig
   ```
3. Look for **IPv4 Address** under the active network adapter.

Output Example:

```
Ethernet adapter Ethernet:
   IPv4 Address: 192.168.1.100
```

2. Finding IP Address on Linux/macOS

Using Terminal:

1. Open the terminal.
2. Type the following command and press Enter:
   ```
   ifconfig
   ```
   (For modern Linux distributions, use `ip a` instead of `ifconfig`.)
3. Look for `inet` under the active network interface (e.g., `eth0` or `wlan0`).

Output Example:

```
wlan0: inet 192.168.1.102
```

3. Finding IP Address of Other Devices in the Network

Using Ping (For Network Devices):

1. Open the terminal or CMD.
2. Type:
   ```
   ping google.com
   ```
3. This will return the IP address of **google.com** (e.g., `142.250.182.78`).

Using ARP (To Find Connected Devices on Local Network):

1. Open CMD or terminal.
2. Run the command:

```
arp -a
```

3. This will list all connected devices along with their IP addresses.

Output Example:

```
192.168.1.1      00-1A-2B-3C-4D-5E    Dynamic
192.168.1.102    00-1B-2C-3D-4E-5F    Dynamic
```

4. Finding IP Address in Python (For Automation in IoT)

```
import socket


hostname = socket.gethostname()

ip_address = socket.gethostbyname(hostname)


print("Device Name:", hostname)

print("IP Address:", ip_address)
```

Output Example:

```
Device Name: MyComputer

IP Address: 192.168.1.105
```

Experiment No. 4:

To Interface LED/Buzzer with Arduino/Raspberry Pi and Write a Program to Turn ON/OFF LED for a Specific Duration

1. Interfacing LED with Arduino

Hardware Required:

- Arduino Uno
- LED
- 220Ω Resistor
- Jumper Wires

Connections:

LED Pin	Arduino Pin
Anode (+)	Digital Pin 13
Cathode (-)	GND

Code for LED Blinking (ON/OFF for 1 Second)

```
void setup() {
    pinMode(13, OUTPUT); // Set pin 13 as output
}

void loop() {
    digitalWrite(13, HIGH); // Turn LED ON
    delay(1000); // Wait for 1 second
    digitalWrite(13, LOW); // Turn LED OFF
    delay(1000); // Wait for 1 second
}
```

Output:

- The LED turns ON for 1 second and then turns OFF for 1 second repeatedly.

2. Interfacing Buzzer with Arduino

Hardware Required:

- Arduino Uno
- Buzzer

- 1kΩ Resistor
- Jumper Wires

Connections:

Buzzer Pin	Arduino Pin
Positive (+)	Digital Pin 9
Negative (-)	GND

Code for Buzzer (ON/OFF for 1 Second)

```
void setup() {
    pinMode(9, OUTPUT); // Set pin 9 as output
}


void loop() {
    digitalWrite(9, HIGH); // Turn Buzzer ON
    delay(1000); // Wait for 1 second
    digitalWrite(9, LOW); // Turn Buzzer OFF
    delay(1000); // Wait for 1 second
}
```

Output:

- The buzzer beeps for 1 second and then stays OFF for 1 second repeatedly.

3. Interfacing LED with Raspberry Pi (Python)

Hardware Required:

- Raspberry Pi
- LED
- 220Ω Resistor
- Breadboard
- Jumper Wires

Connections:

LED Pin	Raspberry Pi GPIO Pin
Anode (+)	GPIO17 (Pin 11)
Cathode (-)	GND (Pin 6)

Python Code for LED Blinking

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

while True:
    GPIO.output(17, GPIO.HIGH)  # Turn LED ON
    time.sleep(1)  # Wait 1 second
    GPIO.output(17, GPIO.LOW)  # Turn LED OFF
    time.sleep(1)  # Wait 1 second
```

Output:

- The LED turns ON and OFF every second.

4. Interfacing Buzzer with Raspberry Pi

Hardware Required:

- Raspberry Pi
- Buzzer
- 1kΩ Resistor
- Jumper Wires

Connections:

Buzzer Pin	Raspberry Pi GPIO Pin
Positive (+)	GPIO18 (Pin 12)
Negative (-)	GND (Pin 6)

Python Code for Buzzer Beep

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
```

```
while True:
    GPIO.output(18, GPIO.HIGH)  # Turn Buzzer ON
    time.sleep(1)  # Wait 1 second
    GPIO.output(18, GPIO.LOW)  # Turn Buzzer OFF
    time.sleep(1)  # Wait 1 second
```

Output:

- The buzzer beeps for 1 second and then turns OFF for 1 second repeatedly.

Experiment No. 5:

To Interface DHT11/DHT22 Sensor with Arduino/Raspberry Pi and Write a Program to Print Temperature and Humidity Readings

1. Interfacing DHT11/DHT22 with Arduino

Hardware Required:

- Arduino Uno
- DHT11 or DHT22 Sensor
- 10kΩ Resistor (for pull-up)
- Jumper Wires

Connections:

DHT11/DHT22 Pin	Arduino Pin
VCC (Pin 1)	5V
Data (Pin 2)	Digital Pin 2
GND (Pin 4)	GND

Install DHT Library:

1. Open **Arduino IDE**.
2. Go to **Sketch** > **Include Library** > **Manage Libraries**.
3. Search for **DHT sensor library** by Adafruit and install it.

Arduino Code to Read Temperature and Humidity

```
#include <DHT.h>
#define DHTPIN 2       // Digital pin connected to DHT sensor
#define DHTTYPE DHT11  // Change to DHT22 if using DHT22


DHT dht(DHTPIN, DHTTYPE);


void setup() {
    Serial.begin(9600);
    dht.begin();
}
```

```
void loop() {
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    if (isnan(temperature) || isnan(humidity)) {
        Serial.println("Failed to read from DHT sensor!");
        return;
    }

    Serial.print("Temperature: ");
    Serial.print(temperature);
    Serial.print("°C  Humidity: ");
    Serial.print(humidity);
    Serial.println("%");

    delay(2000); // Wait 2 seconds before the next reading
}
```

Output (on Serial Monitor):

```
Temperature: 25.3°C  Humidity: 60%
Temperature: 25.4°C  Humidity: 59%
```

2. Interfacing DHT11/DHT22 with Raspberry Pi (Python)

Hardware Required:

- Raspberry Pi
- DHT11 or DHT22 Sensor
- 10kΩ Resistor (for pull-up)
- Jumper Wires

Connections:

DHT11/DHT22 Pin	Raspberry Pi GPIO Pin
VCC (Pin 1)	3.3V (Pin 1)
Data (Pin 2)	GPIO4 (Pin 7)
GND (Pin 4)	GND (Pin 6)

Install Required Libraries:

Run the following commands in the terminal to install dependencies:

```
pip install Adafruit_DHT
```

Python Code to Read Temperature and Humidity

```
import Adafruit_DHT

import time


DHT_SENSOR = Adafruit_DHT.DHT11  # Use DHT22 if needed

DHT_PIN = 4  # GPIO pin connected to the sensor


while True:

    humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)


    if humidity is not None and temperature is not None:

        print(f"Temperature: {temperature:.1f}°C  Humidity: 
{humidity:.1f}%")

    else:

        print("Failed to retrieve data from sensor")


    time.sleep(2)  # Read every 2 seconds
```

Output (on Terminal):

```
Temperature: 24.8°C  Humidity: 58%

Temperature: 25.0°C  Humidity: 57%
```

Experiment No.6:

To Interface PIR Sensor with Arduino/Raspberry Pi and Write a Program to Check Motion Detection

1. Interfacing PIR Sensor with Arduino

Hardware Required:

- Arduino Uno
- PIR Motion Sensor (HC-SR501)
- LED (for indication)
- 220Ω Resistor
- Jumper Wires

Connections:

PIR Sensor Pin	Arduino Pin
VCC (Pin 1)	5V
OUT (Pin 2)	Digital Pin 2
GND (Pin 3)	GND

Arduino Code for Motion Detection

```
#define PIR_SENSOR 2  // PIR sensor output pin
#define LED 13        // LED indicator pin

void setup() {
    pinMode(PIR_SENSOR, INPUT);
    pinMode(LED, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    int motion = digitalRead(PIR_SENSOR);

    if (motion == HIGH) {
        Serial.println("Motion Detected!");
        digitalWrite(LED, HIGH); // Turn ON LED
        delay(2000); // Hold for 2 seconds
    } else {
        digitalWrite(LED, LOW); // Turn OFF LED
    }
}
```

Output (on Serial Monitor):

```
Motion Detected!
```

- The LED turns ON when motion is detected and remains OFF otherwise.

2. Interfacing PIR Sensor with Raspberry Pi (Python)

Hardware Required:

- Raspberry Pi
- PIR Motion Sensor (HC-SR501)
- LED (Optional)
- 220Ω Resistor
- Jumper Wires

Connections:

PIR Sensor Pin	Raspberry Pi GPIO Pin
VCC (Pin 1)	5V (Pin 2)
OUT (Pin 2)	GPIO4 (Pin 7)
GND (Pin 3)	GND (Pin 6)

Python Code for Motion Detection

```
import RPi.GPIO as GPIO
import time

PIR_SENSOR = 4  # GPIO pin connected to PIR sensor

GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_SENSOR, GPIO.IN)

print("PIR Sensor Ready...")
```

```
try:
    while True:
        if GPIO.input(PIR_SENSOR):
            print("Motion Detected!")
            time.sleep(2)  # Wait for stability
        else:
            print("No Motion")
        time.sleep(0.5)

except KeyboardInterrupt:
    print("Program Stopped")
    GPIO.cleanup()
```

Output (on Terminal):

```
PIR Sensor Ready...
Motion Detected!
```

- When motion is detected, "Motion Detected!" is printed.
- If no motion is detected, "No Motion" is displayed.

Experiment No. 7:

To Interface Pi Camera with Arduino/Raspberry Pi and Write a Program to Capture and Save Images

1. Interfacing Pi Camera with Raspberry Pi

Hardware Required:

- Raspberry Pi (Any Model with Camera Port)
- Raspberry Pi Camera Module
- Camera Cable

Connections:

1. Connect the **Pi Camera** to the **CSI (Camera Serial Interface) Port** on the Raspberry Pi.
2. Enable the Camera in Raspberry Pi:
    - Open the terminal and run:
      ```
      sudo raspi-config
      ```
    - Navigate to **Interfacing Options** > **Camera** and enable it.
    - Reboot the Raspberry Pi:
      ```
      sudo reboot
      ```

Python Code to Capture Image and Save to Desktop

```
from picamera import PiCamera
from time import sleep

camera = PiCamera()
camera.resolution = (1024, 768)  # Set resolution

camera.start_preview()
sleep(5)  # Wait for 5 seconds before capturing
image_path = "/home/pi/Desktop/captured_image.jpg"
camera.capture(image_path)  # Save the image to Desktop
camera.stop_preview()

print(f"Image saved at {image_path}")
```

Output:

- The Pi Camera captures an image after 5 seconds and saves it as `captured_image.jpg` on the desktop.

2. Interfacing Pi Camera with Arduino (Using Serial Communication)

Hardware Required:

- Arduino Uno
- Raspberry Pi
- Pi Camera Module
- USB Cable

Concept:

- Arduino cannot directly interface with the Pi Camera, so Raspberry Pi captures the image, and Arduino can trigger it via serial communication.

Arduino Code (Trigger Pi Camera via Serial)

```
void setup() {
    Serial.begin(9600); // Start serial communication
}

void loop() {
    Serial.println("capture"); // Send command to Raspberry Pi
    delay(5000); // Wait before sending the next request
}
```

Python Code on Raspberry Pi (Capturing Image on Command from Arduino)

```
import serial
from picamera import PiCamera
from time import sleep

camera = PiCamera()
camera.resolution = (1024, 768)

# Connect to Arduino (Ensure correct port)
```

```
arduino = serial.Serial('/dev/ttyUSB0', 9600, timeout=1)


while True:
    command = arduino.readline().decode().strip()
    if command == "capture":
        print("Capturing Image...")
        image_path = "/home/pi/Desktop/captured_image.jpg"
        camera.capture(image_path)
        print(f"Image saved at {image_path}")
```

Output:

- When Arduino sends `"capture"`, the Raspberry Pi captures an image and saves it to the desktop.

Experiment No. 8:

To Transmit and Access the Sensed Data to Any Cloud Platform

Objective:

To collect sensor data (e.g., temperature, humidity) using an **Arduino or Raspberry Pi** and transmit it to a cloud platform like **Thingspeak**, **Google Firebase**, or **MQTT broker** for remote monitoring.

1. Using Raspberry Pi to Send Data to Thingspeak

Hardware Required:

- Raspberry Pi
- DHT11/DHT22 Sensor (for temperature & humidity)
- Internet connection (Wi-Fi or Ethernet)

Steps:

1. **Create a Thingspeak Account** at https://thingspeak.com.
2. **Create a New Channel** and note the **Write API Key**.
3. **Install Python Libraries**:

   ```
   pip install requests Adafruit_DHT
   ```

Python Code (Send Data to Thingspeak)

```
import requests

import Adafruit_DHT

import time


API_KEY = "YOUR_THINGSPEAK_WRITE_API_KEY"  # Replace with your API
Key

DHT_SENSOR = Adafruit_DHT.DHT11

DHT_PIN = 4  # GPIO pin for DHT sensor


while True:

    humidity, temperature = Adafruit_DHT.read(DHT_SENSOR, DHT_PIN)
```

```
    if humidity is not None and temperature is not None:
        url = f"https://api.thingspeak.com/update?
api_key={API_KEY}&field1={temperature}&field2={humidity}"
        response = requests.get(url)

        if response.status_code == 200:
            print(f"Data Sent! Temp: {temperature}°C, Humidity:
{humidity}%")
        else:
            print("Failed to send data")

    time.sleep(15)  # Send data every 15 seconds
```

Output (on Thingspeak Cloud Dashboard):

- **Temperature and Humidity Graphs** are updated in real-time.

2. Using Arduino + ESP8266 to Send Data to Firebase

Hardware Required:

- Arduino Uno
- ESP8266 Wi-Fi Module
- DHT11 Sensor

Steps:

1. **Set up Firebase Database** at https://firebase.google.com/.
2. **Get Firebase Database URL and Secret Key**.
3. **Install Required Libraries** in Arduino IDE:
    - `FirebaseESP8266.h`
    - `DHT.h`

Arduino Code (Send Data to Firebase)

```
#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <DHT.h>

#define FIREBASE_HOST "your-firebase-database.firebaseio.com"
```

```
#define FIREBASE_AUTH "your-firebase-secret-key"
#define WIFI_SSID "your-wifi-ssid"
#define WIFI_PASSWORD "your-wifi-password"

#define DHTPIN 2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);

FirebaseData firebaseData;

void setup() {
    Serial.begin(115200);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nConnected to WiFi");
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
    dht.begin();
}

void loop() {
    float temperature = dht.readTemperature();
    float humidity = dht.readHumidity();

    if (!isnan(temperature) && !isnan(humidity)) {
        Firebase.setFloat(firebaseData, "/Temperature",
temperature);
        Firebase.setFloat(firebaseData, "/Humidity", humidity);
        Serial.println("Data sent to Firebase!");
    } else {
        Serial.println("Failed to read from sensor");
    }

    delay(5000);  // Send data every 5 seconds
}
```

Output (on Firebase Cloud Database):

- Temperature and humidity values update in real-time.

Subject Code: LPEIT-101

Subject Name: Business Intelligence & its Applications Laboratory

Programme: B. Tech	**L: 0 T: 0 P: 2**
Semester: 5	**Teaching Hours: 24**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 Hours**
Total Marks: 50	**Course Type: Professional Elective-I**

Detailed Contents:

1. Case Study and Design of a Data Mart Application
2. To study different Data Mining tools
3. To Perform Data Cleaning on Data Sets
4. To Perform association rule mining (AprioriAlgorithm) using any Data Mining tool.
5. To perform classification using Bayes Method using any Data Mining tool.
6. To Perform K-means clustering techniques for data mining on data set using any Data Mining tool.
7. To interpret and visualize the output of data mining using any Data mining tool.

Experiment No. 1

Case Study and Design of a Data Mart Application

Objective:

To understand the concept of a **Data Mart** and design a **Sales Data Mart** for business intelligence and analytics.

1. Introduction to Data Mart

A **Data Mart** is a subset of a **Data Warehouse**, designed for specific business functions like sales, finance, or HR. It stores historical and aggregated data to support **decision-making**.

Types of Data Marts:

1. **Dependent Data Mart** – Extracted from a centralized **Data Warehouse**
2. **Independent Data Mart –** Built from operational databases
3. **Hybrid Data Mart** – Combination of both

2. Case Study: Sales and Customer Analytics Data Mart

Scenario:

A **retail company** wants to analyze **sales, customer purchases, and product demand** across multiple stores. The company needs a **Sales Data Mart** to generate reports for business intelligence.

Data Sources:

- **Sales Transactions** (Sales_ID, Date, Store_ID, Product_ID, Quantity, Revenue)
- **Customer Data** (Customer_ID, Name, Age, Gender, Location)
- **Product Data** (Product_ID, Category, Price, Supplier)

3. Data Mart Design

Fact Table: Sales_Fact

Sales_ID	Date	Store_ID	Product_ID	Customer_ID	Quantity	Revenue
101	2025-03-30	1	501	1001	2	500
102	2025-03-30	2	502	1002	1	300

Dimension Tables:

1. Customer_Dim

Customer_ID	Name	Age	Gender	Location
1001	Abhishek	25	Male	Delhi
1002	Priya	23	Female	Mumbai

2. Product_Dim

Product_ID	Category	Price	Supplier
501	Electronics	250	ABC Ltd.
502	Clothing	300	XYZ Ltd.

3. Store_Dim

Store_ID	Store_Location	Manager
1	Delhi	Mr. Sharma
2	Mumbai	Ms. Kapoor

4. Business Intelligence Insights from Data Mart

- **Sales Trends:** Identify peak sales seasons.
- **Customer Behavior:** Analyze purchases based on age and location.
- **Product Performance:** Determine which products generate the highest revenue.

Experiment No. 2

To Study Different Data Mining Tools

Objective:

To explore various **Data Mining tools** used for extracting patterns, trends, and useful insights from large datasets.

1. Introduction to Data Mining Tools

Data mining tools help in analyzing vast amounts of data to discover hidden patterns and relationships. These tools are used in **business intelligence, fraud detection, market analysis, and healthcare analytics**.

2. Popular Data Mining Tools

1. WEKA (Waikato Environment for Knowledge Analysis)

- Open-source tool developed by the University of Waikato.
- Supports **data preprocessing, classification, clustering, and association rule mining**.
- Used for academic and research purposes.

2. RapidMiner

- A powerful tool for **predictive analytics and machine learning**.
- Supports **drag-and-drop** functionality for easy implementation.
- Used in **finance, healthcare, and retail industries**.

3. KNIME (Konstanz Information Miner)

- An open-source platform for **data preprocessing, machine learning, and visualization**.
- Supports **integrations with Python, R, and big data tools**.
- Widely used in **pharmaceutical and financial analytics**.

4. Orange

- A user-friendly tool with an intuitive **graphical user interface (GUI)**.
- Used for **clustering, classification, and text mining**.
- Preferred for **educational and beginner-level data mining**.

5. Apache Mahout

- A **machine learning library** built for big data applications.
- Works with **Hadoop and Spark** for scalable data mining.
- Used for **recommendation systems and clustering algorithms**.

6. IBM SPSS Modeler

- A commercial tool for **statistical analysis and predictive modeling**.
- Used in **market research, banking, and healthcare**.
- Provides support for **decision trees, neural networks, and regression models**.

7. Tableau

- A **data visualization and business intelligence** tool.
- Allows users to **create dashboards** for better insights.
- Used in **corporate analytics and reporting**.

3. Comparison of Data Mining Tools

Tool	Open Source	Key Features	Industry Usage
WEKA	Yes	Classification, Clustering	Research, Academia
RapidMiner	No	Predictive Analytics, Drag & Drop	Business, Finance
KNIME	Yes	Data Integration, ML	Pharma, Financial
Orange	Yes	GUI-based, Text Mining	Education, Beginners
Apache Mahout	Yes	Scalable ML, Big Data	E-commerce, AI
IBM SPSS Modeler	No	Statistical Analysis	Market Research, Banking
Tableau	No	Data Visualization	Business Intelligence

4. Applications of Data Mining Tools

- **Retail:** Market basket analysis to understand customer behavior.
- **Healthcare:** Disease prediction and patient risk analysis.
- **Finance:** Fraud detection and credit risk analysis.
- **E-commerce:** Recommendation systems for personalized marketing.
- **Social Media:** Sentiment analysis and user behavior tracking.

Experiment No. 3

To Perform Data Cleaning on Data Sets

Objective:

To understand and apply **Data Cleaning** techniques to remove inconsistencies, errors, and missing values from datasets to improve data quality.

1. Introduction to Data Cleaning

Data cleaning is the process of **identifying and correcting inaccurate, incomplete, or inconsistent data** to ensure high-quality and reliable datasets for analysis.

2. Common Data Cleaning Techniques

1. **Handling Missing Values** – Removing or replacing missing entries.
2. **Removing Duplicates** – Identifying and eliminating duplicate records.
3. **Correcting Inconsistencies** – Standardizing inconsistent data formats.
4. **Handling Outliers** – Identifying and removing extreme values that may distort analysis.
5. **Data Type Conversion** – Converting data types for consistency (e.g., String to Date).
6. **Normalization** – Transforming data into a common scale.

3. Steps to Perform Data Cleaning

Step 1: Load the Dataset

- Obtain a raw dataset containing missing values, duplicates, and inconsistencies.

Step 2: Identify Issues in the Dataset

- Check for **null values, duplicate rows, incorrect data formats, and outliers**.

Step 3: Handle Missing Values

- **Remove missing values** if they are insignificant.
- **Replace missing values** with the **mean, median, or mode** for numerical data.
- **Use forward or backward fill** for time-series data.

Step 4: Remove Duplicates

- Identify and **delete duplicate records** to avoid redundancy.

Step 5: Correct Inconsistencies

- Standardize **date formats, spelling errors, or categorical values** (e.g., "Male" vs. "M").

Step 6: Handle Outliers

- Use **statistical techniques like Z-score, IQR (Interquartile Range)** to detect outliers.

- Decide whether to **remove or transform** extreme values.

Step 7: Normalize Data (If Needed)

- Scale numerical values to ensure a uniform range (e.g., Min-Max Scaling).

Step 8: Verify Data Quality

- Perform **final validation** to check for any remaining errors.

4. Example of Data Cleaning Issues and Fixes

Raw Data (Before Cleaning)	Issues	Cleaned Data (After Cleaning)
001, John, 25, Male, NY	No issue	001, John, 25, Male, NY
002, Alice, NA, Female, CA	Missing Value	002, Alice, 28, Female, CA (Filled missing age with median)
003, Mike, 30, M, Texas	Inconsistent Data	003, Mike, 30, Male, Texas
004, Sarah, 22, Female, Florida	No issue	004, Sarah, 22, Female, Florida
004, Sarah, 22, Female, Florida	Duplicate Entry	Removed Duplicate
005, Tom, 500, Male, OH	Outlier in Age	005, Tom, 35, Male, OH (Corrected Age)

5. Applications of Data Cleaning

- **Business Intelligence –** Improves accuracy in decision-making.
- **Healthcare –** Ensures correct patient data for diagnosis.
- **Finance –** Reduces errors in financial transactions.
- **Machine Learning –** Enhances model performance and accuracy.

Experiment No. 4

To Perform Association Rule Mining using Apriori Algorithm in a Data Mining Tool

Objective:

To implement **Association Rule Mining** using the **Apriori Algorithm** in a Data Mining tool and analyze the relationships between items in a dataset.

1. Introduction to Association Rule Mining

- Association Rule Mining is used to **find relationships** between frequently occurring items in large datasets.
- It is widely used in **market basket analysis**, where businesses analyze customer purchase behavior.

2. Apriori Algorithm

- **Apriori** is a popular algorithm for frequent itemset mining and association rule learning.
- It uses the concept of **support, confidence, and lift** to generate meaningful rules.

Key Terms in Apriori Algorithm:

1. **Support:** Frequency of an itemset in the dataset.

   containing A TransactionsSupport(A)=Total TransactionsTransactions containing A
2. **Confidence:** How often the rule is found to be true.

   (A ∩ B) (A)Confidence(A→B)=Support (A)Support (A ∩ B)
3. **Lift:** The strength of a rule compared to random chance.

   (A → B) (B)Lift(A→B)=Support (B)Confidence (A → B)

3. Steps to Perform Apriori Algorithm in a Data Mining Tool

Step 1: Load Dataset

- Use a dataset containing transactions (e.g., a supermarket sales dataset).
- Example transactions:

Transaction ID	Items Purchased
1	Milk, Bread, Butter
2	Milk, Bread
3	Bread, Butter
4	Milk, Butter
5	Bread, Butter, Eggs

Step 2: Apply Apriori Algorithm

- Set **minimum support and confidence thresholds** (e.g., Support = 30%, Confidence = 60%).
- The algorithm finds frequent itemsets and generates rules.

Step 3: Generate Association Rules

Example rules extracted:

Rule	Support	Confidence	Lift
{Milk} → {Bread}	40%	66.7%	1.2
{Bread} → {Butter}	60%	75%	1.5
{Milk, Bread} → {Butter}	30%	80%	1.8

Step 4: Analyze Results

- The rule **{Milk} → {Bread}** suggests that customers who buy **Milk** are likely to buy **Bread**.
- The rule **{Bread} → {Butter}** has **high confidence (75%)**, meaning most customers who buy Bread also buy Butter.

4. Applications of Association Rule Mining

- **Retail & E-commerce:** Market basket analysis to recommend frequently bought products.
- **Healthcare:** Identifying symptoms leading to diseases.
- **Finance:** Detecting fraudulent transactions.
- **Web Analytics:** Understanding user behavior on websites.

Experiment No. 5

To Perform Classification Using Bayes Method in a Data Mining Tool

Objective:

To implement **Bayes Classification** using a Data Mining tool and analyze its effectiveness in classifying data based on probability.

1. Introduction to Bayes Classification

- **Bayes Theorem** is a statistical method for predicting the probability of a class based on prior knowledge.
- It is widely used in **spam filtering, medical diagnosis, and sentiment analysis**.
- The most common application is **Naïve Bayes Classifier**, which assumes that features are independent.

2. Bayes Theorem

The **Bayes Theorem** is expressed as:

$P(A|B)=P(B)P(B|A)\times P(A)$

Where:

- **P(A|B)** = Probability of class **A** given evidence **B** (Posterior Probability).
- **P(B|A)** = Probability of evidence **B** given class **A** (Likelihood).
- **P(A)** = Probability of class **A** (Prior Probability).
- **P(B)** = Probability of evidence **B** (Evidence Probability).

3. Steps to Perform Naïve Bayes Classification in a Data Mining Tool

Step 1: Load Dataset

- Use a dataset with labeled classes (e.g., spam detection dataset).

Email ID	**Words (Feature)**	**Class (Spam/Not Spam)**
1	Free, Offer, Buy	Spam
2	Meeting, Schedule	Not Spam
3	Free, Discount	Spam
4	Project, Update	Not Spam
5	Limited, Buy Now	Spam

Step 2: Apply Bayes Theorem

- Calculate **prior probabilities** of spam and not spam emails.
- Compute **likelihood** of words occurring in spam and not spam emails.
- Apply **Bayes Theorem** to classify new emails.

Step 3: Classify New Data

Example: Classifying **"Free Buy Now"** email.

- Compute **P(Spam | Free, Buy Now)** and **P(Not Spam | Free, Buy Now)**.
- Assign the label to the class with the highest probability.

Step 4: Evaluate Model Performance

- Use **accuracy, precision, recall, and F1-score** to assess classification.

4. Applications of Bayes Classification

- **Spam Detection:** Classifying emails as spam or not.
- **Medical Diagnosis:** Predicting diseases based on symptoms.
- **Sentiment Analysis:** Determining positive or negative sentiments in reviews.
- **Credit Risk Analysis:** Assessing loan default probabilities.

Experiment No. 6

To Perform K-Means Clustering on a Dataset Using a Data Mining Tool

Objective:

To implement **K-Means Clustering** in a Data Mining tool and analyze how data points are grouped into clusters based on similarity.

1. Introduction to K-Means Clustering

- **K-Means** is an **unsupervised learning algorithm** used to **partition data into K clusters**.
- It minimizes the variance within each cluster by grouping similar data points together.
- It is commonly used in **customer segmentation, pattern recognition, and image compression**.

2. Steps in K-Means Clustering Algorithm

1. **Select K (Number of Clusters) –** Choose the number of clusters to form.
2. **Initialize Centroids –** Randomly assign K initial centroids.
3. **Assign Data Points to Nearest Cluster –** Compute the **Euclidean distance** of each point to the nearest centroid.
4. **Recompute Centroids –** Update cluster centers by calculating the mean of all points in a cluster.
5. **Repeat Steps 3 and 4** until centroids do not change significantly.

3. Steps to Perform K-Means Clustering in a Data Mining Tool

Step 1: Load Dataset

- Choose a dataset (e.g., customer data with age, income, spending score).

Customer ID	Age	Income ($)	Spending Score
1	25	50000	80
2	30	60000	40
3	35	70000	20
4	40	80000	90
5	45	90000	50

Step 2: Choose the Value of K

- Use **Elbow Method** to find the optimal number of clusters.

Step 3: Apply K-Means Algorithm

- Randomly select **K centroids**.
- Assign each data point to the nearest centroid.

- Compute new cluster centers and update centroids.

Step 4: Visualize Clusters

- Plot clusters using a **scatter plot** to analyze data separation.

Step 5: Evaluate Clustering Performance

- Use metrics like **inertia (WCSS - Within-Cluster Sum of Squares)** and **Silhouette Score** to measure cluster quality.

4. Applications of K-Means Clustering

- **Customer Segmentation:** Grouping customers based on behavior.
- **Image Segmentation:** Identifying patterns in image pixels.
- **Anomaly Detection:** Detecting unusual patterns in fraud detection.
- **Document Clustering:** Categorizing documents by topics.

Experiment No. 7

To Interpret and Visualize the Output of Data Mining Using Any Data Mining Tool

Objective:

To understand how to interpret and visualize the results obtained from data mining techniques using various visualization tools.

1. Introduction to Data Visualization in Data Mining

- **Data Visualization** helps in understanding patterns, trends, and relationships in data.
- It provides insights through **graphs, charts, heatmaps, and scatter plots**.
- Common visualization methods include:
  - **Histograms** (for frequency distribution)
  - **Scatter Plots** (for clustering analysis)
  - **Box Plots** (for outlier detection)
  - **Bar Charts** (for categorical data analysis)
  - **Heatmaps** (for correlation analysis)

2. Steps to Perform Data Visualization in a Data Mining Tool

Step 1: Load Dataset

- Use a dataset (e.g., customer transactions, stock prices, or healthcare records).

Step 2: Apply a Data Mining Algorithm

- Implement a **classification, clustering, or association rule mining** technique.
- Example: Running **K-Means Clustering** or **Decision Tree Classification**.

Step 3: Generate Visualizations

- Use **charts and graphs** to represent the output of the data mining algorithm.
- Common visualizations include:
  - **Cluster Analysis Visualization** (for K-Means Clustering)
  - **Decision Tree Diagram** (for classification models)
  - **Confusion Matrix** (for model accuracy)
  - **Association Rules Visualization** (for Apriori Algorithm)

Step 4: Interpret Results

- Identify trends and insights from the visual data.

- Example: In a **customer segmentation model**, identify groups based on spending behavior.

3. Example Visualizations

1. Clustering Output (K-Means Clustering)

- **Scatter plot** with different colors for each cluster.

2. Classification Output (Decision Tree)

- **Decision tree diagram** showing classification rules.

3. Association Rule Mining (Apriori Algorithm)

- **Graphical representation of association rules** (e.g., market basket analysis).

4. Accuracy and Performance Metrics

- **Confusion Matrix** for model evaluation.
- **ROC Curve** to compare different classifiers.

4. Applications of Data Visualization in Data Mining

- **Business Intelligence:** Understanding sales trends and customer behavior.
- **Healthcare Analytics:** Identifying disease patterns.
- **Fraud Detection:** Detecting anomalies in transactions.
- **Stock Market Analysis:** Predicting stock price movements.

Subject Code: LPEIT-102

Subject Name: Advanced Web Technologies Laboratory

Programme: B. Tech	**L: 0 T: 0 P: 2**
Semester: 5	**Teaching Hours: 24**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 Hours**
Total Marks: 50	**Course Type: Professional Elective-I**

Detailed Contents:
1. To install and setup the HTML5 based Bootstrap framework and to deploy basic HTML elements using Bootstrap CSS.
2. To understand and deploy the multicolumn grid layout of Bootstrap.
3. To deploy different types of buttons, progress bars, modals and navigation bars using Bootstrap.
4. To create and setup the Git repository on Bitbucket or github using SSH.
5. To perform push, c lone and patch operation to Git repository.
6. To install and setup the CodeIgniter Framework and to understand its MVC architecture.
7. To construct a simple login page web application to authenticate users using CodeIgniter Framework and also perform CURD operations.
8. To install and setup, configure the Laravel Framework.
9. To construct the any simple web application using Laravel Framework.

Experiment No. 1

To Install and Setup the HTML5 Based Bootstrap Framework and Deploy Basic HTML Elements Using Bootstrap CSS

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <title>Bootstrap Setup</title>

    <!-- Bootstrap CSS CDN -->

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">

</head>

<body>


    <div class="container text-center mt-5">

        <h1 class="text-primary">Welcome to Bootstrap</h1>

        <p class="lead">This is a simple Bootstrap setup with
basic HTML elements.</p>


        <button class="btn btn-success">Click Me</button>


        <div class="alert alert-info mt-3" role="alert">

            Bootstrap is successfully installed and working!

        </div>

    </div>


    <!-- Bootstrap JS CDN -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>
```

Output:

- A **centered page** with a **heading**, **paragraph**, and a **Bootstrap-styled button**.
- An **alert box** confirming that Bootstrap is installed successfully.
- Responsive design with Bootstrap's built-in classes.

Experiment No. 2

To Understand and Deploy the Multicolumn Grid Layout of Bootstrap

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <title>Bootstrap Grid Layout</title>

    <!-- Bootstrap CSS CDN -->

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">

</head>

<body>


    <div class="container mt-4">

        <h2 class="text-center text-primary">Bootstrap Multicolumn
Grid Layout</h2>


        <div class="row mt-4">

            <div class="col-md-4 bg-light p-3 border">

                <h4>Column 1</h4>

                <p>This is the first column in a 3-column
layout.</p>

            </div>

            <div class="col-md-4 bg-secondary text-white p-3
border">

                <h4>Column 2</h4>

                <p>This is the second column in a 3-column
layout.</p>
```

```
            </div>
            <div class="col-md-4 bg-light p-3 border">
                <h4>Column 3</h4>
                <p>This is the third column in a 3-column
layout.</p>
            </div>
        </div>

        <div class="row mt-4">
            <div class="col-md-6 bg-warning p-3 border">
                <h4>Column 1 (Half Width)</h4>
                <p>This column takes half the screen width.</p>
            </div>
            <div class="col-md-6 bg-success text-white p-3
border">
                <h4>Column 2 (Half Width)</h4>
                <p>This column takes the other half.</p>
            </div>
        </div>
    </div>

    <!-- Bootstrap JS CDN -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>
</body>
</html>
```

Output:

- A **responsive grid layout** using Bootstrap.
- **First row**: Three equal-width columns.
- **Second row**: Two equal-width columns (each taking half of the screen).
- **Different background colors** for easy differentiation.
- The layout **adjusts on smaller screens** automatically.

Experiment No. 3

To Deploy Different Types of Buttons, Progress Bars, Modals, and Navigation Bars Using Bootstrap

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <title>Bootstrap Components</title>

    <!-- Bootstrap CSS CDN -->

    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootst
rap.min.css" rel="stylesheet">

</head>

<body>


    <!-- Navigation Bar -->

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

        <div class="container-fluid">

            <a class="navbar-brand" href="#">My Website</a>

            <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav">

                    <li class="nav-item"><a class="nav-link
active" href="#">Home</a></li>

                    <li class="nav-item"><a class="nav-link"
href="#">About</a></li>
```

```
                    <li class="nav-item"><a class="nav-link"
href="#">Contact</a></li>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <!-- Buttons -->
        <h3 class="text-primary">Bootstrap Buttons</h3>
        <button class="btn btn-primary">Primary</button>
        <button class="btn btn-success">Success</button>
        <button class="btn btn-danger">Danger</button>
        <button class="btn btn-warning">Warning</button>

        <!-- Progress Bar -->
        <h3 class="text-primary mt-4">Progress Bar</h3>
        <div class="progress">
            <div class="progress-bar progress-bar-striped bg-
success" role="progressbar" style="width: 70%;">70%</div>
        </div>

        <!-- Modal Trigger Button -->
        <h3 class="text-primary mt-4">Modal Example</h3>
        <button type="button" class="btn btn-info" data-bs-
toggle="modal" data-bs-target="#myModal">Open Modal</button>

        <!-- Modal -->
        <div class="modal fade" id="myModal" tabindex="-1" aria-
labelledby="modalLabel" aria-hidden="true">
            <div class="modal-dialog">
                <div class="modal-content">
                    <div class="modal-header">
```

```
                        <h5 class="modal-title"
id="modalLabel">Bootstrap Modal</h5>

                        <button type="button" class="btn-close"
data-bs-dismiss="modal"></button>

                    </div>

                    <div class="modal-body">

                        This is a Bootstrap modal pop-up.

                    </div>

                    <div class="modal-footer">

                        <button type="button" class="btn btn-
secondary" data-bs-dismiss="modal">Close</button>

                    </div>

                </div>

            </div>

        </div>

    </div>


    <!-- Bootstrap JS CDN -->

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstra
p.bundle.min.js"></script>

</body>

</html>
```

Output:

- A **dark navigation bar** with links for Home, About, and Contact.
- **Different types of buttons** (Primary, Success, Danger, Warning).
- **A progress bar** displaying 70% progress.
- **A modal pop-up** that opens when clicking the "Open Modal" button.

Experiment No. 4

To Create and Setup the Git Repository on Bitbucket or GitHub Using SSH

Steps to Perform the Experiment:

1. Generate an SSH Key (If Not Already Generated)

Open the terminal and run:

```
ssh-keygen -t rsa -b 4096 -C "your-email@example.com"
```

- Press **Enter** to save the key at the default location (`~/.ssh/id_rsa`).
- Enter a passphrase (optional).

2. Add SSH Key to SSH Agent

Start the SSH agent:

```
eval "$(ssh-agent -s)"
```

Add the private key to the agent:

```
ssh-add ~/.ssh/id_rsa
```

3. Copy SSH Key

Run:

```
cat ~/.ssh/id_rsa.pub
```

Copy the output (your public SSH key).

4. Add SSH Key to GitHub/Bitbucket

- **GitHub:** Go to **GitHub** → **Settings** → **SSH and GPG keys** → **New SSH Key**, paste the key, and save.
- **Bitbucket:** Go to **Bitbucket** → **Personal Settings** → **SSH Keys** → **Add Key**, paste the key, and save.

5. Verify SSH Connection

Run:

```
ssh -T git@github.com
```

or

```
ssh -T git@bitbucket.org
```

If successful, it will display a welcome message.

6. Create a Git Repository and Push Code

- Initialize a repository:

  ```
  git init
  ```

- Add a remote repository:

  ```
  git remote add origin git@github.com:username/repository.git
  ```

  (Replace `username/repository.git` with your actual repository URL.)

- Create a sample file:

  ```
  echo "# My Project" > README.md
  git add README.md
  git commit -m "Initial commit"
  git push -u origin main
  ```

Output:

- SSH key successfully added to GitHub/Bitbucket.
- Git repository initialized and code pushed using SSH.
- Running `ssh -T git@github.com` confirms authentication.

Experiment No. 5

To Perform Push, Clone, and Patch Operation to Git Repository

Steps to Perform the Experiment

1. Push Operation (Uploading Local Changes to Remote Repository)

Step 1: Initialize a Git Repository (If Not Already Done)

```
git init
```

Step 2: Add Remote Repository

```
git remote add origin git@github.com:username/repository.git
```

(Replace `username/repository.git` with your actual repository URL.)

Step 3: Add and Commit Files

```
echo "Hello Git" > file.txt
git add file.txt
git commit -m "Added file.txt"
```

Step 4: Push Code to GitHub/Bitbucket

```
git push -u origin main
```

Output of Push Operation:

- The repository gets updated with the new file on GitHub/Bitbucket.

2. Clone Operation (Copying a Remote Repository to Local System)

Step 1: Clone a Repository

```
git clone git@github.com:username/repository.git
```

(Replace `username/repository.git` with your actual repository URL.)

Step 2: Verify Cloned Repository

```
cd repository
ls
```

(This should list all the files from the remote repository.)

Output of Clone Operation:

- The complete repository is copied to the local machine.

3. Patch Operation (Creating and Applying a Patch File)

Step 1: Modify a File

```
echo "New line added" >> file.txt
```

Step 2: Create a Patch File

```
git diff > my_patch.patch
```

(This saves the changes in `my_patch.patch`.)

Step 3: Apply the Patch to Another Repository

```
git apply my_patch.patch
```

Output of Patch Operation:

- The changes stored in the patch file are successfully applied to the repository.

Experiment No. 6

To Install and Setup the CodeIgniter Framework and Understand Its MVC Architecture

Steps to Perform the Experiment

1. Install and Setup CodeIgniter Framework

Step 1: Download CodeIgniter

- Go to the official website: CodeIgniter Download
- Download the latest version of CodeIgniter.
- Extract the downloaded folder to your web server's root directory (`htdocs` for XAMPP or `www` for WAMP).

Step 2: Configure Base URL

- Open the `config.php` file inside `application/config/`:

```
$config['base_url'] = 'http://localhost/codeigniter/';
```

Step 3: Configure Autoload Settings

- Open `autoload.php` inside `application/config/` and update:

```
$autoload['libraries'] = array('database', 'session');
$autoload['helper'] = array('url', 'form');
```

Step 4: Set Up Routes

- Open `routes.php` inside `application/config/` and set the default controller:

```
$route['default_controller'] = 'Welcome';
```

Step 5: Run CodeIgniter

- Start your web server (Apache for XAMPP/WAMP).
- Open a browser and visit:

```
http://localhost/codeigniter/
```

- You should see the default welcome page of CodeIgniter.

2. Understanding CodeIgniter MVC Architecture

Model-View-Controller (MVC) Architecture in CodeIgniter:

- **Model**: Manages database operations.
- **View**: Handles UI (HTML, CSS, JavaScript).
- **Controller**: Acts as an intermediary between Model and View.

Example of MVC Implementation in CodeIgniter

Step 1: Create a Model (`application/models/UserModel.php`)

```
<?php
class UserModel extends CI_Model {
    public function getUsers() {
        return array("Alice", "Bob", "Charlie");
    }
}
?>
```

Step 2: Create a Controller (`application/controllers/UserController.php`)

```
<?php
class UserController extends CI_Controller {
    public function index() {
        $this->load->model('UserModel');
        $data['users'] = $this->UserModel->getUsers();
        $this->load->view('user_view', $data);
    }
}
?>
```

Step 3: Create a View (`application/views/user_view.php`)

```
<!DOCTYPE html>
<html>
<head>
    <title>Users List</title>
</head>
<body>
    <h2>List of Users:</h2>
    <ul>
        <?php foreach ($users as $user) { ?>
            <li><?php echo $user; ?></li>
        <?php } ?>
    </ul>
```

```
</body>

</html>
```

Step 4: Configure Route (`application/config/routes.php`)

```
$route['users'] = 'UserController';
```

Step 5: Run the Application

Visit:

```
http://localhost/codeigniter/index.php/users
```

This will display the list of users fetched from the model.

Output:

- CodeIgniter is successfully installed and set up.
- MVC structure is implemented correctly.
- Visiting `/users` in the browser displays a list of users.

Experiment No. 7

To Construct a Simple Login Page Web Application to Authenticate Users Using CodeIgniter Framework and Perform CRUD Operations

1. Setup CodeIgniter Framework

Follow the installation steps from the previous experiment to install and configure CodeIgniter.

2. Database Setup

Step 1: Create a Database

Create a database named `ci_auth`.

Step 2: Create a Users Table

Run the following SQL query in MySQL:

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL
);
```

3. Configure Database in CodeIgniter

Edit `application/config/database.php`:

```
$db['default'] = array(
    'dsn'   => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'ci_auth',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
);
```

4. Create the Model (`UserModel.php`)

Create `application/models/UserModel.php`:

```
<?php
class UserModel extends CI_Model {

    public function register($data) {
        return $this->db->insert('users', $data);
    }

    public function login($username, $password) {
        $this->db->where('username', $username);
        $query = $this->db->get('users');
        $user = $query->row();

        if ($user && password_verify($password, $user->password))
{
            return $user;
        }
        return false;
    }

    public function getUsers() {
        return $this->db->get('users')->result();
    }

    public function deleteUser($id) {
        return $this->db->delete('users', array('id' => $id));
    }
}
```

```
?>
```

5. Create the Controller (`AuthController.php`)

Create `application/controllers/AuthController.php`:

```
<?php
class AuthController extends CI_Controller {

    public function __construct() {
        parent::__construct();
        $this->load->model('UserModel');
        $this->load->library('session');
    }

    public function register() {
        $this->load->view('register_view');
    }

    public function registerUser() {
        $data = array(
            'username' => $this->input->post('username'),
            'password' => password_hash($this->input-
>post('password'), PASSWORD_DEFAULT),
            'email' => $this->input->post('email')
        );

        if ($this->UserModel->register($data)) {
            redirect('AuthController/login');
        }
    }

    public function login() {
        $this->load->view('login_view');
    }

    public function loginUser() {
        $username = $this->input->post('username');
        $password = $this->input->post('password');

        $user = $this->UserModel->login($username, $password);

        if ($user) {
            $this->session->set_userdata('user', $user);
            redirect('AuthController/dashboard');
        } else {
            echo "Invalid Credentials!";
        }
    }

    public function dashboard() {
        if (!$this->session->userdata('user')) {
            redirect('AuthController/login');
        }
        $data['users'] = $this->UserModel->getUsers();
        $this->load->view('dashboard_view', $data);
    }

    public function logout() {
        $this->session->unset_userdata('user');
        redirect('AuthController/login');
    }

    public function deleteUser($id) {
        $this->UserModel->deleteUser($id);
        redirect('AuthController/dashboard');
```

```
    }
}
?>
```

6. Create the Views

Register View (`register_view.php`)

Create `application/views/register_view.php`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Register</title>
</head>
<body>
    <h2>Register</h2>
    <form method="post" action="<?= 
base_url('AuthController/registerUser') ?>">
        <input type="text" name="username" placeholder="Username" 
required><br>
        <input type="email" name="email" placeholder="Email" 
required><br>
        <input type="password" name="password" 
placeholder="Password" required><br>
        <button type="submit">Register</button>
    </form>
</body>
</html>
```

Login View (`login_view.php`)

Create `application/views/login_view.php`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
```

```
    <h2>Login</h2>

    <form method="post" action="<?=
base_url('AuthController/loginUser') ?>">

        <input type="text" name="username" placeholder="Username"
required><br>

        <input type="password" name="password"
placeholder="Password" required><br>

        <button type="submit">Login</button>

    </form>

</body>

</html>
```

Dashboard View (`dashboard_view.php`)

Create `application/views/dashboard_view.php`:

```
<!DOCTYPE html>

<html>

<head>

    <title>Dashboard</title>

</head>

<body>

    <h2>Welcome, <?= $this->session->userdata('user')->username; ?
></h2>

    <h3>Registered Users:</h3>

    <ul>

        <?php foreach ($users as $user) { ?>

            <li><?= $user->username; ?> - <a href="<?=
base_url('AuthController/deleteUser/'.$user->id)
?>">Delete</a></li>

        <?php } ?>

    </ul>

    <br>

    <a href="<?= base_url('AuthController/logout') ?>">Logout</a>

</body>

</html>
```

7. Configure Routes (`application/config/routes.php`)

```
$route['register'] = 'AuthController/register';
$route['login'] = 'AuthController/login';
$route['dashboard'] = 'AuthController/dashboard';
```

8. Run the Application

Start the server and visit:

- **Register Page:** `http://localhost/codeigniter/index.php/register`
- **Login Page:** `http://localhost/codeigniter/index.php/login`
- **Dashboard (after login):** `http://localhost/codeigniter/index.php/dashboard`

Output:

1. **User Registration**: A new user is successfully stored in the database.
2. **User Login**: Authenticated users can log in and access the dashboard.
3. **CRUD Operations**: Users can view and delete registered users from the dashboard.
4. **Logout Functionality**: Users can log out securely.

Experiment No. 8:

To Install, Setup, and Configure the Laravel Framework

Step 1: Install Laravel Framework

Prerequisites:

- PHP (>= 8.0)
- Composer (PHP Dependency Manager)
- MySQL or SQLite for database
- A web server (Apache/Nginx)

Installation Steps:

1. **Install Composer**
   If you haven't installed Composer, download and install it from:
   https://getcomposer.org/download/
2. **Install Laravel via Composer**
   Open the terminal and run:
   ```
   composer create-project --prefer-dist laravel/laravel my_laravel_app
   ```
   This command will create a Laravel project inside a folder named `my_laravel_app`.
3. **Move into the Project Directory**
   ```
   cd my_laravel_app
   ```
4. **Start the Laravel Development Server**
   ```
   php artisan serve
   ```
   This will start a development server, and you can access Laravel at:
   ```
   http://127.0.0.1:8000
   ```

Step 2: Configure Laravel

1. Configure Environment File (`.env`)

Open the `.env` file in the root directory and configure the database settings:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=my_laravel_db
DB_USERNAME=root
DB_PASSWORD=
```

(Modify database credentials as per your local setup.)

Run the following command to generate the application key:

```
php artisan key:generate
```

2. Configure Database Migration

To set up the database, run:

```
php artisan migrate
```

This will create necessary tables in the database.

3. Setup Laravel Routes

Modify `routes/web.php` to define a sample route:

```
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});
```

4. Create a Controller

Run the following command to create a new controller:

```
php artisan make:controller TestController
```

Modify `app/Http/Controllers/TestController.php`:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TestController extends Controller
{
    public function index() {
        return "Laravel is set up successfully!";
    }
}
?>
```

Define a route to access the controller in `routes/web.php`:

```
Route::get('/test', [TestController::class, 'index']);
```

Now, visiting `http://127.0.0.1:8000/test` should display:
"Laravel is set up successfully!"

Step 3: Running the Laravel Application

1. Start the Laravel server:

   ```
   php artisan serve
   ```

2. Open `http://127.0.0.1:8000/` in a browser.

Output:

1. Laravel successfully installed and running.
2. Database connection configured.
3. A sample controller is working correctly.

Experiment No. 9

To Construct a Simple Web Application Using Laravel Framework

Objective:

To develop a simple **Task Management Web Application** using the Laravel framework. The application will allow users to **add, view, edit, and delete tasks**.

Step 1: Install Laravel Framework

If Laravel is not installed, run the following command:

```
composer create-project --prefer-dist laravel/laravel TaskManager
```

Navigate to the project directory:

```
cd TaskManager
```

Start the Laravel development server:

```
php artisan serve
```

Access the application at:

```
http://127.0.0.1:8000
```

Step 2: Configure Database

Modify the `.env` file to set up the database connection:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=task_manager
DB_USERNAME=root
DB_PASSWORD=
```

Create the database manually in MySQL:

```
CREATE DATABASE task_manager;
```

Run Laravel migrations:

```
php artisan migrate
```

Step 3: Create a Model, Migration & Controller

1. **Generate Model & Controller**

   ```
   php artisan make:model Task -mcr
   ```

   This creates a **Task model**, a **migration file**, and a **controller**.

2. **Define Database Schema (Migration File: `database/migrations/xxxx_xx_xx_create_tasks_table.php`)**
   Modify the `up()` function in the migration file:

```
public function up()

{

    Schema::create('tasks', function (Blueprint $table) {

        $table->id();

        $table->string('title');

        $table->text('description')->nullable();

        $table->boolean('completed')->default(false);

        $table->timestamps();

    });

}
```

   Run the migration:

```
php artisan migrate
```

Step 4: Define Routes (`routes/web.php`)

```
use App\Http\Controllers\TaskController;

Route::get('/', [TaskController::class, 'index']);
Route::post('/tasks', [TaskController::class, 'store']);
Route::get('/tasks/{id}/edit', [TaskController::class, 'edit']);
Route::post('/tasks/{id}', [TaskController::class, 'update']);
Route::delete('/tasks/{id}', [TaskController::class, 'destroy']);
```

Step 5: Implement TaskController (`app/Http/Controllers/TaskController.php`)

```
<?php


namespace App\Http\Controllers;


use Illuminate\Http\Request;

use App\Models\Task;


class TaskController extends Controller

{

    public function index()
    {
        $tasks = Task::all();
        return view('tasks.index', compact('tasks'));
    }

    public function store(Request $request)
    {
        Task::create($request->all());
        return redirect('/');
    }

    public function edit($id)
    {
        $task = Task::find($id);
        return view('tasks.edit', compact('task'));
    }

    public function update(Request $request, $id)
    {
        $task = Task::find($id);
        $task->update($request->all());
        return redirect('/');
    }

    public function destroy($id)
    {
        Task::destroy($id);
        return redirect('/');
    }
}
```

Step 6: Create Views

1. Task List Page (`resources/views/tasks/index.blade.php`)

```
<!DOCTYPE html>
<html>
<head>
    <title>Task Manager</title>
</head>
<body>
    <h2>Task List</h2>
    <form action="/tasks" method="POST">
        @csrf
        <input type="text" name="title" placeholder="Task Title">
        <input type="text" name="description" 
placeholder="Description">
        <button type="submit">Add Task</button>
    </form>

    <ul>
        @foreach($tasks as $task)
            <li>{{ $task->title }} - {{ $task->description }}
                <a href="/tasks/{{ $task->id }}/edit">Edit</a>
                <form action="/tasks/{{ $task->id }}" 
method="POST" style="display:inline;">
                    @csrf
                    @method('DELETE')
                    <button type="submit">Delete</button>
                </form>
            </li>
        @endforeach
    </ul>
</body>
</html>
```

2. Edit Task Page (`resources/views/tasks/edit.blade.php`)

```
<!DOCTYPE html>
<html>
<head>
    <title>Edit Task</title>
</head>
<body>
    <h2>Edit Task</h2>
    <form action="/tasks/{{ $task->id }}" method="POST">
        @csrf
        <input type="text" name="title" value="{{ $task->title }}">
        <input type="text" name="description" value="{{ $task->description }}">
        <button type="submit">Update Task</button>
    </form>
</body>
</html>
```

Step 7: Run the Application

1. Start the Laravel server:
   ```
   php artisan serve
   ```
2. Open `http://127.0.0.1:8000/` in a browser.
3. Add, edit, or delete tasks.

Output:

- A simple **Task Management Web Application** is created.
- Users can **add, view, edit, and delete tasks**.
- Data is stored in a MySQL database.

Subject Code: LPEIT-104

Subject Name: Software Modelling and Analysis Laboratory

Programme: B. Tech	**L: 0 T: 0 P: 2**
Semester: 5	**Teaching Hours: 24**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 Hours**
Total Marks: 50	**Course Type: Professional Elective-I**

Detailed Contents:

1. Identify a software system that needs to be developed and prepare a software requirement specification for that system.
2. Identify use cases and develop the use-case model for the system.
3. Identify the classes and develop the class diagrams for the system.
4. Find the interactions between the objects and represent those using UML sequence and collaboration diagrams.
5. Draw the relevant state chart diagrams and activity diagrams for the system.
6. Implement the system as per the detailed design by using an object-oriented programming language.
7. Test all the scenario is developed as per the use-case diagram.
8. Perform failure modes and effect analysis and fault-tree analysis of the system.
9. Check the model and improve its reusability and maintainability.
10. Implement the modified system and test it for various scenarios

Experiment No. 1:

Identify a Software System and Prepare a Software Requirement Specification

Software System: Library Management System (LMS)

1. Introduction

- **Purpose:** The Library Management System (LMS) aims to automate the management of books, users, and transactions within a library.
- **Scope:** This system allows students and librarians to manage book issues, returns, fines, and book searches efficiently.
- **Users:** Librarians, Students, and Administrators.

2. Functional Requirements

- **User Authentication:** Login and registration for users.
- **Book Management:** Add, update, delete, and search books.
- **Student Management:** Maintain student details and issue limits.
- **Issue & Return System:** Record book issues and returns with due dates.
- **Fine Calculation:** Automatically calculate fines for late returns.
- **Reports & Logs:** Generate reports for issued books and overdue books.

3. Non-Functional Requirements

- **Performance:** The system should handle up to 1000 concurrent users.
- **Security:** User authentication with encryption for passwords.
- **Availability:** 99.9% uptime with database backups.

4. System Models

- **Use Case Diagram:** Shows interactions between librarians, students, and the system.
- **ER Diagram:** Defines relationships between books, students, and transactions.
- **DFD (Data Flow Diagram):** Illustrates how data moves through the system.

5. Assumptions & Constraints

- The system will run on a web-based platform.
- Data storage will be handled by a MySQL database.
- Users must have unique login credentials.

Experiment No. 2:

Identify Use Cases and Develop the Use-Case Model for the System

System: Library Management System (LMS)

1. Identified Use Cases

Use Case ID	Use Case Name	Actors	Description
UC01	User Authentication	Librarian, Student	Allows users to log in or register into the system.
UC02	Add New Books	Librarian	Enables the librarian to add new books to the library database.
UC03	Search Books	Librarian, Student	Allows users to search for available books in the library.
UC04	Issue Books	Librarian	Issues a book to a student and records the due date.
UC05	Return Books	Librarian, Student	Processes the return of books and updates the system.
UC06	Calculate Fine	Librarian	Computes the fine if the book is returned late.
UC07	Generate Reports	Librarian	Generates reports on issued, returned, and overdue books.

2. Use-Case Diagram for Library Management System

Actors:

- **Librarian**: Manages books, issues/returns books, and calculates fines.
- **Student**: Searches for books and requests for book issuance/return.

Use-Case Diagram (Description)

- The **Librarian** can **log in**, **add books**, **issue books**, **return books**, **calculate fines**, and **generate reports**.
- The **Student** can **log in**, **search books**, **borrow books**, and **return books**.

Experiment No. 3:

Identify the Classes and Develop the Class Diagram for the System

1. Identified Classes for Library Management System (LMS)

Class Name	Attributes	Methods	Description
User	userID, name, email, password	login(), logout(), register()	Represents a general user in the system.
Librarian	employeeID, designation	addBook(), issueBook(), returnBook(), calculateFine()	Inherits from User, performs book transactions.
Student	studentID, course	searchBook(), requestIssue(), returnBook()	Inherits from User, requests book issue/return.
Book	bookID, title, author, category, status	checkAvailability(), updateStatus()	Represents a book in the library.
Transaction	transactionID, studentID, bookID, issueDate, returnDate, fineAmount	issueBook(), returnBook(), calculateFine()	Manages book lending and fine calculations.
Report	reportID, reportType, generatedDate	generateReport()	Generates reports on issued/returned books.

2. Class Diagram for Library Management System

Relationships Between Classes:

1. **Inheritance:**
   - `Librarian` and `Student` inherit from `User`.
2. **Association:**
   - `Librarian` can **issue** and **return** `Books`.
   - `Student` can **search** and **borrow** `Books`.
   - `Transaction` links `Student` and `Book` with issue/return details.
   - `Report` is generated based on `Transaction` details.

Experiment No. 4:

Find the Interactions Between Objects and Represent Using UML Sequence and Collaboration Diagrams

1. Identified Interactions for Library Management System (LMS)

Use Case: Issue Book

- **Actors:** Student, Librarian
- **Objects Involved:** Student, Librarian, Book, Transaction

Steps in Interaction:

1. Student requests to issue a book.
2. Librarian checks book availability.
3. If available, the librarian issues the book and records the transaction.
4. Book status is updated to “Issued.”
5. Confirmation is sent to the student.

2. UML Sequence Diagram for Book Issue

Objects:

- **Student** (Initiates request)
- **Librarian** (Processes request)
- **Book** (Verifies availability)
- **Transaction** (Records the issue)

Sequence of Interactions:

1. Student → Librarian: **Request for book issue**
2. Librarian → Book: **Check availability**
3. Book → Librarian: **Confirm availability**
4. Librarian → Transaction: **Record book issue details**
5. Transaction → Book: **Update book status**
6. Librarian → Student: **Confirm book issued**

3. UML Collaboration Diagram for Book Issue

- **Student** interacts with **Librarian**.
- **Librarian** checks with **Book** for availability.
- **Librarian** updates **Transaction** records.
- **Book** status is updated based on the transaction.

Experiment No. 5:

Draw the Relevant State Chart Diagrams and Activity Diagrams for the System

1. State Chart Diagram for Book Object

States of a Book in Library Management System (LMS)

- **Available** → The book is available in the library.
- **Requested** → A student requests to issue the book.
- **Issued** → The book is issued to a student.
- **Returned** → The student returns the book.
- **Overdue** → The return date is exceeded, and a fine is applicable.

State Transitions:

1. **Available** → (Student requests) → **Requested**
2. **Requested** → (Librarian issues book) → **Issued**
3. **Issued** → (Student returns book) → **Returned**
4. **Issued** → (Due date crossed) → **Overdue**
5. **Overdue** → (Student pays fine & returns book) → **Returned**

2. Activity Diagram for Book Issue Process

Actors Involved: Student, Librarian, System

Steps in the Activity Diagram:

1. Student searches for a book.
2. Student requests book issue.
3. Librarian checks book availability.
4. If the book is available, it is issued to the student.
5. The system updates the transaction record.
6. Book status is changed to **Issued**.
7. Student receives a confirmation.

Experiment No. 6

Implement the System as per the Detailed Design Using an Object-Oriented Programming Language

Implementation Overview:

The **Library Management System (LMS)** will be implemented using an **Object-Oriented Programming Language** such as **Java, C++, or Python**. It will follow the **OOP principles** (Encapsulation, Inheritance, Polymorphism, and Abstraction) to structure the system based on the class diagrams and use-case models.

System Components:

1. **Classes Implemented:**
   - `User` (Base class)
   - `Librarian` (Inherits from `User`)
   - `Student` (Inherits from `User`)
   - `Book` (Handles book-related operations)
   - `Transaction` (Manages book issuing & returning)
   - `Report` (Generates system reports)
2. **Key Functionalities:**
   - **User Authentication** (Login/Logout)
   - **Book Management** (Adding, Searching, Issuing, Returning Books)
   - **Transaction Handling** (Recording Issued & Returned Books)
   - **Fine Calculation** (For Overdue Books)
   - **Report Generation** (Issued/Returned Books Report)

Experiment No. 7:

Test All the Scenarios Developed as per the Use-Case Diagram

Testing the Library Management System (LMS)

The system is tested to ensure that all scenarios function as described in the **Use-Case Diagram**. The test cases cover **functional, boundary, and error handling scenarios**.

1. Test Scenarios & Expected Outputs

Test Case ID	Scenario	Expected Output	Status
TC1	User Login (Valid Credentials)	Successful Login	Passed
TC2	User Login (Invalid Credentials)	Error: Invalid Username/Password	Passed
TC3	Add a New Book	Book Successfully Added	Passed
TC4	Search for an Existing Book	Book Found with Details	Passed
TC5	Search for a Non-Existing Book	Error: Book Not Found	Passed
TC6	Student Requests to Issue a Book	Request Sent to Librarian	Passed
TC7	Librarian Issues a Book	Book Status Updated to "Issued"	Passed
TC8	Student Returns a Book	Book Status Updated to "Available"	Passed
TC9	Student Returns Book After Due Date	Fine Calculated and Displayed	Passed
TC10	Generate Report of Issued Books	List of Issued Books Displayed	Passed

Experiment No. 8:

Perform Failure Modes and Effect Analysis and Fault-Tree Analysis of the System

1. Failure Modes and Effect Analysis (FMEA)

Objective: Identify possible failure modes in the **Library Management System (LMS)** and analyze their effects.

Failure Mode	Possible Cause	Effect on System	Severity (1-10)	Detection Probability (1-10)	Action Taken
User unable to log in	Incorrect password input	Access Denied	5	8	Implement password reset option
System crashes during book issue	Database connection failure	Book transaction failure	9	6	Ensure database redundancy
Book not found in search	Incorrect book entry	Frustration for users	4	7	Implement error handling and book catalog update
Fine not calculated for overdue books	Incorrect date tracking	Financial loss	7	5	Improve date tracking algorithm
Unauthorized access to system	Weak authentication	Data security risk	10	4	Implement two-factor authentication

2. Fault-Tree Analysis (FTA)

Top-Level Event: System Failure in LMS

Main Causes and Breakdown:

1. **Authentication Failure**
    - User enters incorrect credentials
    - Database connection timeout
    - Weak encryption of passwords
2. **Database Failure**
    - Server crashes
    - Corrupt database entries
    - No backup mechanism

3. **Book Issuance Failure**
    - Book status not updated correctly
    - Transaction log missing
    - System crash during processing
4. **Fine Calculation Failure**
    - Incorrect date handling
    - No overdue book tracking
    - Incorrect fine rules

Experiment No. 9:

Check the Model and Improve Its Reusability and Maintainability

1. Analysis of the Current Model

Issues Identified in the Library Management System (LMS):

1. **Code Duplication:** Similar functionalities (e.g., book search and book issue) may be repeated.
2. **Tightly Coupled Modules:** The system lacks modularity, making updates difficult.
3. **Limited Scalability:** The system may not handle a large number of users efficiently.
4. **Hardcoded Data:** Book categories, fine amounts, and user roles should be configurable.
5. **Poor Exception Handling:** Lack of structured error handling for system failures.

2. Strategies for Improving Reusability & Maintainability

Improvement Strategy	Implementation Approach	Benefits
Modular Design	Separate `User`, `Book`, and `Transaction` classes	Increases reusability and reduces complexity
Use of Inheritance & Polymorphism	Create a base `User` class and extend `Librarian` & `Student`	Reduces redundant code
Database Normalization	Normalize tables to eliminate data redundancy	Enhances maintainability and data consistency
Configuration Files	Store fine rates, book categories, and admin settings in config files	Allows easy updates without modifying code
Exception Handling	Implement `try-catch` blocks for database and I/O operations	Prevents system crashes
Logging Mechanism	Introduce logging to track system issues	Simplifies debugging and maintenance
API-Based Architecture	Use REST APIs for book search and issue operations	Enables integration with mobile/web applications

Experiment No. 10:

Implement the Modified System and Test It for Various Scenarios

1. Implementation of the Modified System

After improving the **Library Management System (LMS)** for better reusability and maintainability, the following modifications were implemented:

Key Modifications:

Modular Design: Separated classes for `User`, `Book`, `Transaction`, and `DatabaseHandler`.
Object-Oriented Approach: Used **inheritance** (`User → Librarian` and `Student`) and **polymorphism**.
Database Normalization: Ensured proper table structures for books, users, and transactions.
Exception Handling: Added structured `try-catch` blocks to prevent system crashes.
Logging Mechanism: Integrated logging to record errors and actions for debugging.
API-Based Operations: Created **REST APIs** for book transactions, making it extendable for web and mobile apps.

2. Test Scenarios & Expected Outputs

Test Case ID	Scenario	Expected Output	Status
TC1	User Login (Valid Credentials)	Successfully logs in	Passed
TC2	User Login (Invalid Credentials)	Displays "Invalid Username/Password"	Passed
TC3	Add a New Book	Book added successfully	Passed
TC4	Search for an Existing Book	Book found with details	Passed
TC5	Search for a Non-Existing Book	Displays "Book Not Found"	Passed
TC6	Issue a Book (Available)	Updates book status to "Issued"	Passed
TC7	Issue a Book (Already Issued)	Displays "Book Not Available"	Passed
TC8	Return a Book (Before Due Date)	Updates book status to "Available"	Passed
TC9	Return a Book (After Due Date)	Calculates and displays fine	Passed
TC10	Generate Reports (Issued Books)	Displays a list of issued books	Passed
TC11	Unauthorized Access Attempt	Access Denied with error log	Passed
TC12	Server Down Simulation	System logs error and prevents crash	Passed

Subject Code: LPEIT-105

Subject Name: Cryptography Laboratory

Programme: B. Tech	**L: 0 T: 0 P: 2**
Semester: 5	**Teaching Hours: 24**
Theory/Practical: Practical	**Credits: 1**
Internal Marks: 30	**Percentage of Numerical/Design Problems: 100%**
External Marks: 20	**Duration of End Semester Exam(ESE): 1.5 Hours**
Total Marks: 50	**Course Type: Professional Elective-I**

Detailed Contents:

1. Write a C program that contains a string (char pointer) with a value \HelloWorld’. The program should XOR each character in this string with 0 and displays the result.
2. Write a C program that contains a string (char pointer) with a value \HelloWorld’. The program should AND, OR and XOR each character in this string with 127 and display the result.
3. Write a Java program to perform encryption and decryption using the following algorithms: Ceaser Cipher , Substitution Cipher, Hill Cipher.
4. Write a Java program to implement the DES algorithm logic
5. Write a C/JAVA program to implement the Blow Fish algorithm logic
6. Write a C/JAVA program to implement Euclidean Algorithm
7. Using Java Cryptography, encrypt the text “Hello world” using Blowfish. Create your own key using Java keytool.
8. Write a Java program to implement RSA Algorithm
9. Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).
10. Calculate the message digest of a text using the Secure Hash Algorithm (SHA-1)-1 algorithm in JAVA.
11. Calculate the message digest of a text using the Message-Digest algorithm (MD5) algorithm in JAVA

Experiment No. 1:

XOR Operation on String Characters

Code:

```
#include <stdio.h>


int main() {
    char str[] = "HelloWorld";


    printf("Original String after XOR with 0: ");
    for (int i = 0; str[i] != '\0'; i++) {
        printf("%c", str[i] ^ 0); // XOR with 0 keeps the
character unchanged
    }


    printf("\n");
    return 0;
}
```

Output:

```
HelloWorld
```

Experiment No. 2:

Bitwise AND, OR, and XOR Operations on String Characters

Code:

```
#include <stdio.h>


int main() {
    char str[] = "HelloWorld";

    printf("Original String: %s\n", str);

    // AND operation with 127
    printf("After AND with 127: ");
    for (int i = 0; str[i] != '\0'; i++) {
        printf("%c", str[i] & 127);
    }
    printf("\n");

    // OR operation with 127
    printf("After OR with 127: ");
    for (int i = 0; str[i] != '\0'; i++) {
        printf("%c", str[i] | 127);
    }
    printf("\n");

    // XOR operation with 127
    printf("After XOR with 127: ");
    for (int i = 0; str[i] != '\0'; i++) {
        printf("%c", str[i] ^ 127);
    }
    printf("\n");
    return 0;
}
```

Output:

```
Original String: HelloWorld
After AND with 127: HelloWorld
After OR with 127: 
After XOR with 127: '
```

Experiment No. 3:

Encryption and Decryption using Caesar Cipher, Substitution Cipher, and Hill Cipher

Code:

```
import java.util.*;


public class CipherAlgorithms {


    // Caesar Cipher Encryption
    public static String caesarCipherEncrypt(String text, int
shift) {
        StringBuilder result = new StringBuilder();
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                char base = Character.isUpperCase(c) ? 'A' : 'a';
                result.append((char) ((c - base + shift) % 26 +
base));
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }


    // Caesar Cipher Decryption
    public static String caesarCipherDecrypt(String text, int
shift) {
        return caesarCipherEncrypt(text, 26 - shift);
    }


    // Substitution Cipher Encryption
    public static String substitutionCipherEncrypt(String text,
String key) {
        String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        text = text.toUpperCase();
        StringBuilder result = new StringBuilder();
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                int index = alphabet.indexOf(c);
                result.append(key.charAt(index));
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }

    // Substitution Cipher Decryption
    public static String substitutionCipherDecrypt(String text,
String key) {
        String alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        text = text.toUpperCase();
        StringBuilder result = new StringBuilder();
        for (char c : text.toCharArray()) {
            if (Character.isLetter(c)) {
                int index = key.indexOf(c);
                result.append(alphabet.charAt(index));
            } else {
                result.append(c);
            }
        }
        return result.toString();
    }

    // Hill Cipher Encryption (2x2 Key Matrix)
    public static String hillCipherEncrypt(String text, int[][]
keyMatrix) {
        text = text.toUpperCase().replaceAll("[^A-Z]", ""); //
Removing non-alphabetic characters
        if (text.length() % 2 != 0) {
            text += "X"; // Padding if necessary
        }

        StringBuilder result = new StringBuilder();
        for (int i = 0; i < text.length(); i += 2) {
            int[] vector = {text.charAt(i) - 'A', text.charAt(i +
1) - 'A'};
            int[] encryptedVector = {
                (keyMatrix[0][0] * vector[0] + keyMatrix[0][1] *
vector[1]) % 26,
                (keyMatrix[1][0] * vector[0] + keyMatrix[1][1] *
vector[1]) % 26
            };
            result.append((char) (encryptedVector[0] + 'A'));
            result.append((char) (encryptedVector[1] + 'A'));
        }
        return result.toString();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Caesar Cipher
        System.out.println("Enter text for Caesar Cipher:");
        String text = scanner.nextLine();
        System.out.println("Enter shift value:");
        int shift = scanner.nextInt();
        scanner.nextLine();
        String caesarEncrypted = caesarCipherEncrypt(text, shift);
        System.out.println("Caesar Cipher Encrypted: " +
caesarEncrypted);
        System.out.println("Caesar Cipher Decrypted: " +
caesarCipherDecrypt(caesarEncrypted, shift));


        // Substitution Cipher
        System.out.println("Enter text for Substitution Cipher:");
        text = scanner.nextLine();
        System.out.println("Enter 26-letter substitution key:");
        String key = scanner.nextLine().toUpperCase();
        String substitutionEncrypted =
substitutionCipherEncrypt(text, key);
        System.out.println("Substitution Cipher Encrypted: " +
substitutionEncrypted);
        System.out.println("Substitution Cipher Decrypted: " +
substitutionCipherDecrypt(substitutionEncrypted, key));


        // Hill Cipher
        int[][] keyMatrix = {{6, 24}, {1, 13}}; // Example 2x2
matrix
        System.out.println("Enter text for Hill Cipher (length
must be even):");
        text = scanner.nextLine();
        String hillEncrypted = hillCipherEncrypt(text, keyMatrix);
        System.out.println("Hill Cipher Encrypted: " +
hillEncrypted);


        scanner.close();
    }
}
```

Output:

```
Enter text for Caesar Cipher:
HELLO
Enter shift value:
3
Caesar Cipher Encrypted: KHOOR
Caesar Cipher Decrypted: HELLO

Enter text for Substitution Cipher:
HELLO
Enter 26-letter substitution key:
QWERTYUIOPASDFGHJKLZXCVBNM
Substitution Cipher Encrypted: ITSSG
Substitution Cipher Decrypted: HELLO


Enter text for Hill Cipher (length must be even):
TEST
Hill Cipher Encrypted: ZEBB
```

Experiment No. 4:

Implementation of DES Algorithm in Java

Code:

```
import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import java.util.Base64;


public class DESAlgorithm {

    public static void main(String[] args) {

        try {

            // Generate a DES key

            KeyGenerator keyGenerator =
KeyGenerator.getInstance("DES");

            SecretKey secretKey = keyGenerator.generateKey();


            // Sample plaintext message

            String plainText = "HELLODES";


            // Encrypt the message

            Cipher cipher = Cipher.getInstance("DES");

            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

            byte[] encryptedBytes =
cipher.doFinal(plainText.getBytes());

            String encryptedText =
Base64.getEncoder().encodeToString(encryptedBytes);

            System.out.println("Encrypted Text: " +
encryptedText);


            // Decrypt the message

            cipher.init(Cipher.DECRYPT_MODE, secretKey);

            byte[] decryptedBytes =
cipher.doFinal(Base64.getDecoder().decode(encryptedText));
```

```
            String decryptedText = new String(decryptedBytes);
            System.out.println("Decrypted Text: " +
decryptedText);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Encrypted Text: [Some Encrypted Data in Base64]
Decrypted Text: HELLODES
```

Experiment No. 5:

Implementation of Blowfish Algorithm in Java

Code:

```
import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;

import javax.crypto.SecretKey;

import java.util.Base64;


public class BlowfishAlgorithm {

    public static void main(String[] args) {

        try {

            // Generate a Blowfish key

            KeyGenerator keyGenerator =
KeyGenerator.getInstance("Blowfish");

            SecretKey secretKey = keyGenerator.generateKey();


            // Sample plaintext message

            String plainText = "HELLOBLOWFISH";


            // Encrypt the message

            Cipher cipher = Cipher.getInstance("Blowfish");

            cipher.init(Cipher.ENCRYPT_MODE, secretKey);

            byte[] encryptedBytes =
cipher.doFinal(plainText.getBytes());

            String encryptedText =
Base64.getEncoder().encodeToString(encryptedBytes);

            System.out.println("Encrypted Text: " +
encryptedText);


            // Decrypt the message

            cipher.init(Cipher.DECRYPT_MODE, secretKey);

            byte[] decryptedBytes =
cipher.doFinal(Base64.getDecoder().decode(encryptedText));
```

```
            String decryptedText = new String(decryptedBytes);
            System.out.println("Decrypted Text: " +
decryptedText);


        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Encrypted Text: [Some Encrypted Data in Base64]
Decrypted Text: HELLOBLOWFISH
```

Experiment No. 6:

Implementation of Euclidean Algorithm in C and Java

C Program:

```c
#include <stdio.h>

// Function to compute GCD using Euclidean Algorithm
int euclideanGCD(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main() {
    int num1, num2;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    printf("GCD of %d and %d is: %d\n", num1, num2,
euclideanGCD(num1, num2));

    return 0;
}
```

Output:

```
Enter two numbers: 56 98
GCD of 56 and 98 is: 14
```

Java Program:

```java
import java.util.Scanner;


public class EuclideanAlgorithm {
    // Function to compute GCD using Euclidean Algorithm
    public static int euclideanGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter two numbers: ");
        int num1 = scanner.nextInt();
        int num2 = scanner.nextInt();


        System.out.println("GCD of " + num1 + " and " + num2 + "
is: " + euclideanGCD(num1, num2));


        scanner.close();
    }
}
```

Output:

```
Enter two numbers: 56 98
GCD of 56 and 98 is: 14
```

Experiment No. 7:

Encryption of "Hello World" using Blowfish with Java Cryptography and Custom Key

Step 1: Generate a Key using Java KeyTool

Run the following command in the terminal to generate a Blowfish key:

```
keytool -genseckey -alias myBlowfishKey -keyalg Blowfish -keysize 128 -storetype
JCEKS -keystore mykeystore.jceks
```

- `-genseckey` → Generates a secret key.
- `-alias myBlowfishKey` → Sets the alias name of the key.
- `-keyalg Blowfish` → Specifies Blowfish as the encryption algorithm.
- `-keysize 128` → Specifies a key size of 128 bits.
- `-storetype JCEKS` → Java Cryptography Extension KeyStore format.
- `-keystore mykeystore.jceks` → Saves the key in a keystore file.

Step 2: Java Program to Encrypt and Decrypt "Hello World" using Blowfish

```
import javax.crypto.Cipher;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.SecretKeySpec;

import java.io.FileInputStream;

import java.security.KeyStore;

import java.util.Base64;


public class BlowfishEncryption {

    public static void main(String[] args) {

        try {

            // Load the keystore

            FileInputStream fis = new
FileInputStream("mykeystore.jceks");

            KeyStore keystore = KeyStore.getInstance("JCEKS");

            keystore.load(fis, "mypassword".toCharArray()); //
Keystore password
```

```
    }

            // Retrieve the secret key
            SecretKey secretKey = (SecretKey)
keystore.getKey("myBlowfishKey", "mypassword".toCharArray()); //
Key password

            // Plain text to encrypt
            String plainText = "Hello world";

            // Encrypt the text
            Cipher cipher = Cipher.getInstance("Blowfish");
            cipher.init(Cipher.ENCRYPT_MODE, secretKey);
            byte[] encryptedBytes =
cipher.doFinal(plainText.getBytes());
            String encryptedText =
Base64.getEncoder().encodeToString(encryptedBytes);
            System.out.println("Encrypted Text: " +
encryptedText);

            // Decrypt the text
            cipher.init(Cipher.DECRYPT_MODE, secretKey);
            byte[] decryptedBytes =
cipher.doFinal(Base64.getDecoder().decode(encryptedText));
            String decryptedText = new String(decryptedBytes);
            System.out.println("Decrypted Text: " +
decryptedText);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Encrypted Text: [Some Encrypted Data in Base64]
Decrypted Text: Hello world
```

Experiment No. 8:

Implementation of RSA Algorithm in Java

Code:

```
import java.security.*;

import javax.crypto.Cipher;

import java.util.Base64;


public class RSAAlgorithm {

    public static void main(String[] args) {

        try {

            // Generate RSA key pair

            KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA");

            keyPairGen.initialize(2048);

            KeyPair keyPair = keyPairGen.generateKeyPair();

            PublicKey publicKey = keyPair.getPublic();

            PrivateKey privateKey = keyPair.getPrivate();


            // Sample plaintext message

            String plainText = "HELLO RSA";


            // Encrypt the message

            Cipher cipher = Cipher.getInstance("RSA");

            cipher.init(Cipher.ENCRYPT_MODE, publicKey);

            byte[] encryptedBytes =
cipher.doFinal(plainText.getBytes());

            String encryptedText =
Base64.getEncoder().encodeToString(encryptedBytes);

            System.out.println("Encrypted Text: " +
encryptedText);


            // Decrypt the message
            cipher.init(Cipher.DECRYPT_MODE, privateKey);
            byte[] decryptedBytes =
cipher.doFinal(Base64.getDecoder().decode(encryptedText));
            String decryptedText = new String(decryptedBytes);
            System.out.println("Decrypted Text: " +
decryptedText);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
Encrypted Text: [Some Encrypted Data in Base64]
Decrypted Text: HELLO RSA
```

Experiment No. 9:

Implementation of Diffie-Hellman Key Exchange using HTML and JavaScript

Code:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Diffie-Hellman Key Exchange</title>

    <script>

        function generateKeys() {

            // Public parameters (agreed by both parties)

            const p = 23;  // Prime number

            const g = 5;   // Primitive root


            // Alice chooses a private key

            let alicePrivateKey = Math.floor(Math.random() * 10) +
1;

            let alicePublicKey = Math.pow(g, alicePrivateKey) % p;


            // Bob chooses a private key

            let bobPrivateKey = Math.floor(Math.random() * 10) +
1;

            let bobPublicKey = Math.pow(g, bobPrivateKey) % p;


            // Exchange public keys and compute shared secret

            let aliceSharedSecret = Math.pow(bobPublicKey,
alicePrivateKey) % p;

            let bobSharedSecret = Math.pow(alicePublicKey,
bobPrivateKey) % p;
```

```
            // Display results
            document.getElementById("output").innerHTML = `
                <strong>Public Parameters:</strong> p = ${p}, g =
${g}<br>
                <strong>Alice's Private Key:</strong> $
{alicePrivateKey}<br>
                <strong>Alice's Public Key:</strong> $
{alicePublicKey}<br>
                <strong>Bob's Private Key:</strong> $
{bobPrivateKey}<br>
                <strong>Bob's Public Key:</strong> ${bobPublicKey}
<br>
                <strong>Alice's Shared Secret:</strong> $
{aliceSharedSecret}<br>
                <strong>Bob's Shared Secret:</strong> $
{bobSharedSecret}<br>
                <strong>Keys Match:</strong> ${aliceSharedSecret
=== bobSharedSecret ? "Yes" : "No"}
            `;
        }
    </script>
</head>
<body>
    <h2>Diffie-Hellman Key Exchange</h2>
    <button onclick="generateKeys()">Generate Keys</button>
    <div id="output"></div>
</body>
</html>
```

Output (Example Run):

```
Public Parameters: p = 23, g = 5

Alice's Private Key: 6

Alice's Public Key: 8

Bob's Private Key: 15

Bob's Public Key: 19

Alice's Shared Secret: 2

Bob's Shared Secret: 2

Keys Match: Yes
```

Experiment No. 10:

Message Digest Calculation using SHA-1 in Java

Code:

```
import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;


public class SHA1Hashing {

    public static void main(String[] args) {

        try {

            // Get user input

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter text to hash: ");

            String text = scanner.nextLine();

            scanner.close();


            // Create MessageDigest instance for SHA-1

            MessageDigest md = MessageDigest.getInstance("SHA-1");


            // Compute the hash

            byte[] hashBytes = md.digest(text.getBytes());


            // Convert byte array to hexadecimal representation

            StringBuilder hashHex = new StringBuilder();

            for (byte b : hashBytes) {

                hashHex.append(String.format("%02x", b));

            }


            // Display output

            System.out.println("SHA-1 Hash: " +
hashHex.toString());
```

```
        } catch (NoSuchAlgorithmException e) {
            System.out.println("SHA-1 Algorithm not found.");
            e.printStackTrace();
        }
    }
}
```

Output Example:

```
Enter text to hash: HelloWorld
SHA-1 Hash: 2ef7bde608ce5404e97d5f042f95f89f1c232871
```

Experiment No. 11:

Message Digest Calculation using MD5 in Java

Code:

```
import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.util.Scanner;


public class MD5Hashing {

    public static void main(String[] args) {

        try {

            // Get user input

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter text to hash: ");

            String text = scanner.nextLine();

            scanner.close();


            // Create MessageDigest instance for MD5

            MessageDigest md = MessageDigest.getInstance("MD5");


            // Compute the hash

            byte[] hashBytes = md.digest(text.getBytes());


            // Convert byte array to hexadecimal representation

            StringBuilder hashHex = new StringBuilder();

            for (byte b : hashBytes) {

                hashHex.append(String.format("%02x", b));

            }


            // Display output

            System.out.println("MD5 Hash: " + hashHex.toString());
        } catch (NoSuchAlgorithmException e) {
            System.out.println("MD5 Algorithm not found.");
            e.printStackTrace();
        }
    }
}
```

Output Example:

```
Enter text to hash: HelloWorld
MD5 Hash: fc3ff98e8c6a0d3087d515c0473f8677
```