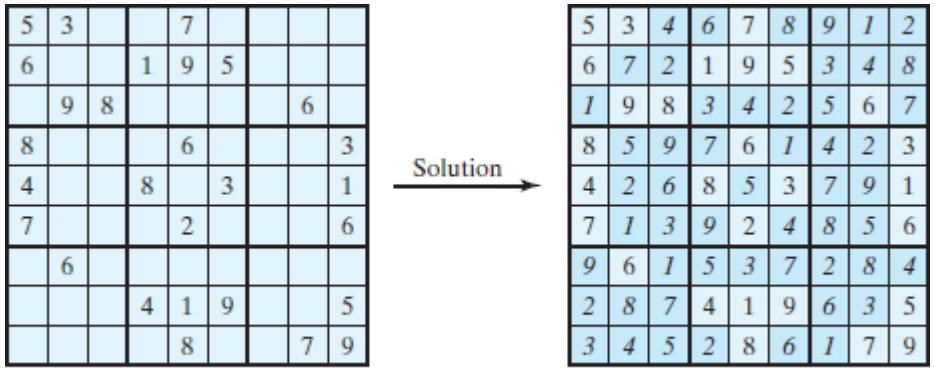
Practical List

PART-I Data Types, Variables, Arrays, Operators, Control Statements, String		LO	PO	PEO
1.	Introduction to Object Oriented Concepts, comparison of Java with other object oriented programming languages. Introduction to JDK, JRE, JVM, javadoc, command line argument	1	1,3, 4,5	1,7
2.	Given a string, return a string made of the first 2 chars (if present), however include first char only if it is 'o' and include the second only if it is 'z', so "ozymandias" yields "oz". startOz("ozymandias") → "oz" startOz("bzoo") → "z" startOz("oxx") → "o"	1	1,3, 4,5	7
3.	Given two non-negative int values, return true if they have the same last digit, such as with 27 and 57. Note that the % "mod" operator computes remainders, so 17 % 10 is 7. lastDigit(7, 17) → true lastDigit(6, 17) → false lastDigit(3, 113) → true	1	1,3, 4,5	7
4.	Given an array of ints, return true if the sequence of numbers 1, 2, 3 appears in the array somewhere. array123([1, 1, 2, 3, 1]) → true array123([1, 1, 2, 4, 1]) → false array123([1, 1, 2, 1, 2, 3]) → true	1	1,3, 4,5	7
5.	Given 2 strings, a and b, return the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings. stringMatch("xxcaazz", "xxbaaz") → 3 stringMatch("abc", "abc") → 2 stringMatch("abc", "axc") → 0	1	1,3, 4,5	7
6.	Given an array of strings, return a new array without the strings that are equal to the target string. One approach is to count the occurrences of the target string, make a new array of the correct length, and then copy over the correct strings. wordsWithout(["a", "b", "c", "a"], "a") → ["b", "c"] wordsWithout(["a", "b", "c", "a"], "b") → ["a", "c", "a"] wordsWithout(["a", "b", "c", "a"], "c") → ["a", "b", "a"]	1	1,3, 4,5	7

7.	Display pyramid.			
	1 1 2 1 1 2 4 2 1 1 2 4 8 4 2 1 1 2 4 8 16 8 4 2 1 1 2 4 8 16 32 16 8 4 2 1 1 2 4 8 16 32 64 32 16 8 4 2 1 1 2 4 8 16 32 64 128 64 32 16 8 4 2 1		4,5	
8.	The problem is to write a program that will grade multiple-choice tests. Assume there are eight students and ten questions, and the answers are stored in a two-dimensional array. Each row records a student's answers to the questions, as shown in the following array. Students' Answers to the Questions: 0 1 2 3 4 5 6 7 8 9 Student 0 A B A C C D E E A D Student 1 D B A B C A E E A D Student 2 E D D A C B E E A D Student 3 C B A E D C E E A D Student 4 A B D C C D E E A D Student 5 B B E C C D E E A D Student 6 B B A C C D E E A D Student 7 E B E C C D E E A D The key is stored in a one-dimensional array: Key to the Questions: 0 1 2 3 4 5 6 7 8 9 Key D B D C C D A E A D Your program grades the test and displays the result. It compares each student's answers with the key, counts the number of correct answers, and displays it.	1	1,3, 4,5	7

9.	**The problem is to check whether a given Sudoku solution is correct.**  (a) Puzzle (b) Solution	1	1,3, 4,5	7
10.	Implement Caesar Cipher.	1	1,3, 4,5	7

PART-II Object Oriented Programming : Classes, Methods, Inheritance				
1.	Design a class named Circle containing following attributes and behavior. • One double data field named radius. The default value is 1. • A no-argument constructor that creates a default circle. • A Single argument constructor that creates a Circle with the specified radius. • A method named getArea() that returns area of the Circle. • A method named getPerimeter() that returns perimeter of it.	1,3	1,3, 4,5 6	3
2.	Design a class named Account that contains: • A private int data field named id for the account (default 0). • A private double data field named balance for the account (default 500₹). • A private double data field named annualInterestRate that stores the current interest rate (default 7%). Assume all accounts have the same interest rate. • A private Date data field named dateCreated that stores the date when the account was created. • A no-arg constructor that creates a default account. • A constructor that creates an account with the specified id and initial balance. • The accessor and mutator methods for id, balance, and annualInterestRate. • The accessor method for dateCreated. • A method named getMonthlyInterestRate() that returns the monthly interest rate. • A method named getMonthlyInterest() that returns the monthly interest. • A method named withdraw that withdraws a specified amount from the account. • A method named deposit that deposits a specified amount to the account.	1,3	1,3, 4,5 6	3
3.	**Use the Account class created as above to simulate an ATM machine.** **Create 10 accounts with id AC001…..AC010 with initial balance 300₹. The system prompts the users to enter an id. If the id is entered incorrectly, ask the user to enter a correct id. Once an id is accepted, display menu with multiple choices.** **1. Balance inquiry** **2. Withdraw money [Maintain minimum balance 300₹]** **3. Deposit money** **4. Money Transfer** **5. Create Account** **6. Deactivate Account** **7. Exit** **Hint**: Use **ArrayList**, which is can shrink and expand with compared to Array.	1,3	1,3, 4,5 6	3
4.	(Subclasses of Account) In Programming Exercise 2, the Account class was defined to model a bank account. An account has the properties account number, balance, annual interest rate, and date created, and methods to deposit and withdraw funds. Create two subclasses for checking and saving accounts. A checking account has an overdraft limit, but a savings account cannot be overdrawn. Draw the UML diagram for the classes and then implement them. Write a test program that creates objects of Account, SavingsAccount, and CheckingAccount and invokes their toString() methods.	1,3	1,3, 4,5 6	3
5.	Develop a Program that illustrate method overloading concept.	1,3	1,3, 4,5 6	3

PART-III

Package & Interface				
1.	WAP that illustrate the use of interface reference. Interface Luminious Object has two method lightOn() and lightOff(). There is one class Solid extended by 2 classes Cube and Cone. There is one class LuminiousCone extends Cone and implements Luminoius Interface. LumminuiousCube extends Cube and implements Luminious Interface. Create a object of LuminiousCone and LuminousCube and use the concept of interface reference to invoke the methods of interface.	3	1,4,5 6	3
2.	WAP that illustrate the interface inheritance. Interface P is extended by P1 and P2 interfaces. Interface P12 extends both P1 and P2. Each interface declares one method and one constant. Create one class that implemetns P12. By using the object of the class invokes each of its method and displays constant.	3	1,4,5 6	3
3.	Create an abstract class Robot that has the concretre subclasses , RobotA, RobotB, RobotC. Class RobotA1 extends RobotA, RobotB1 extends RobotB and RobotC1 extends RobotC. There is interface Motion that declares 3 methods forward(), reverse() and stop(), implemented by RobotB and RobotC. Sound interface declare method beep() implemented by RobotA1, RobotB1 and RobotC1. Create an instance method of each class and invoke beep() and stop() method by all objects.	3	1,4,5 6	3
4.	Develop a Program that illustrate method overriding concept.	3	1,4,5 6	3
5.	Write a java program which shows importing of classes from other user define packages.	3	1,4,5 6	3
6.	Write a program that demonstrates use of packages & import statements.	3	1,4,5 6	3
7.	Write a program that illustrates the significance of interface default method.	3	1,4,5 6	3

PART-IV Exception Handling				
1.	WAP to show the try - catch block to catch the different types of exception.	3	1,4,5 6	3
2.	WAP to generate user defined exception using “throw” and “throws” keyword.	3	1,4,5 6	3
3.	Write a program that raises two exceptions. Specify two ‘catch’ clauses for the two exceptions. Each ‘catch’ block handles a different type of exception. For example the exception could be ‘ArithmeticException’ and ‘ArrayIndexOutOfBoundsException’. Display a message in the ‘finally’ block.	3	1,4,5 6	3

PART-V File Handling & Streams				
1.	WAP to show how to create a file with different mode and methods of File class to find path, directory etc.	2	2,3, 4,5	3,8
2.	Write a program to show a tree view of files and directories under a specified drive/volume.	2	2,3,	3,8
3.	Write a program to transfer data from one file to another file so that if the destination file does not exist, it is created.	2	4,5	3,8
4.	WAP to show use of character and byte stream.	2	2,3,	3,8
5.	WAP to read console input and write them into a file. (BufferedReader /BufferedWriter).	2	4,5	3,8
6.	WAP to demonstrate methods of wrapper class.	2	2,3,	3,8

PART-VI Multithreading				
1.	Write a program to create thread which display "Hello World" message. A. by extending Thread class B. by using Runnable interface.	2	2,3, 4,5	3,8
2.	Write a program which takes N and number of threads as an argument. Program should distribute the task of summation of N numbers amongst number of threads and final result to be displayed on the console.	2	2,3, 4,5	3,8
3.	Write a program to increment the value of one variable by one and display it after one second using thread using sleep() method.	2	2,3, 4,5	3,8
4.	Write a program to create three threads 'FIRST', 'SECOND', 'THIRD'. Set the priority of the 'FIRST' thread to 3, the 'SECOND' thread to 5(default) and the 'THIRD' thread to 7.	2	2,3, 4,5	3,8
5.	**Write a program to solve producer-consumer problem using thread synchronization.**	2	2,3, 4,5	3,8

PART-VII Collection Framework and Generic				
1.	Create a generic method for sorting an array of Comparable objects.	2	2,3, 4,5	3,8
2.	**Write a program that counts the occurrences of words in a text and displays the words and their occurrences in alphabetical order of the words. Using Map and Set Classes.**	2	2,3, 4,5	3,8
3.	**Write a code which counts the number of the keywords in a Java source file. Store all the keywords in a HashSet and use the contains method to test if a word is in the keyword set.**	2	2,3, 4,5	3,8