Name -Rishu Raj

Class/Sec – 631-B

UID-22BCS15617

Subject – Java

Ques:1-

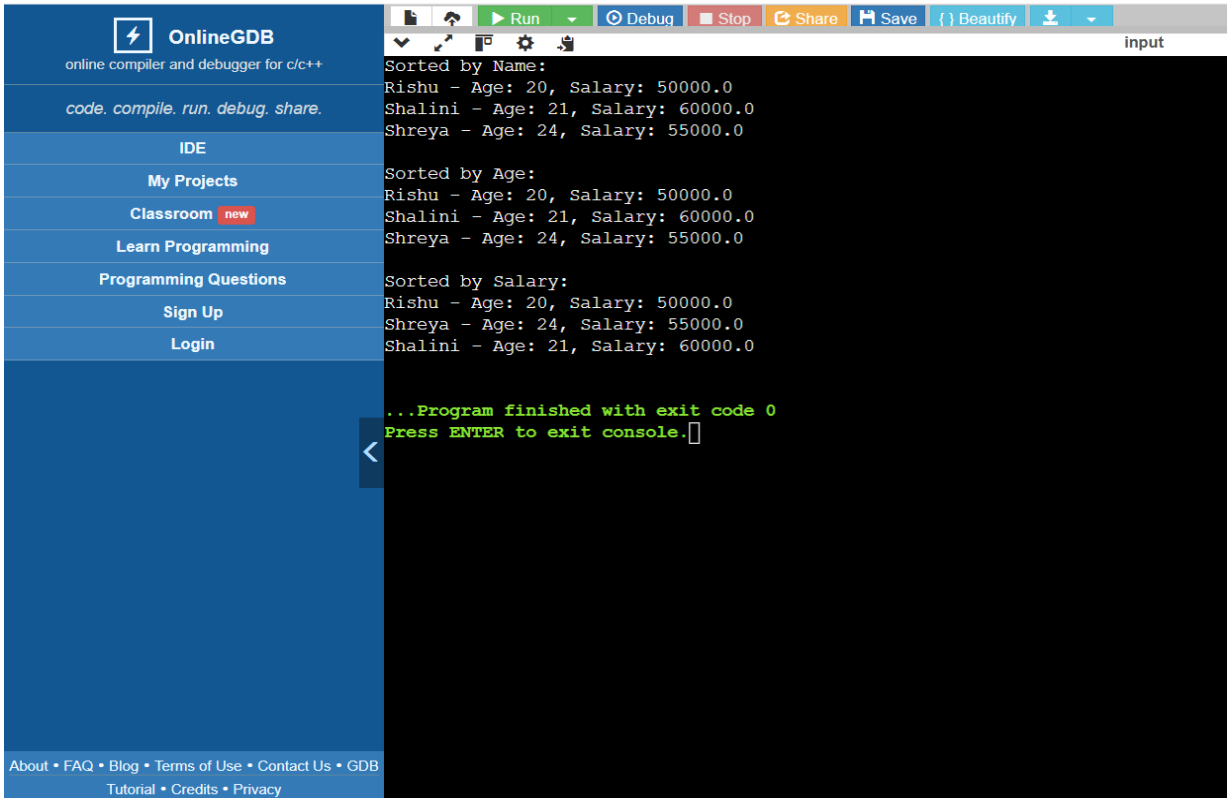
1.Write a program to sort a list of Employee objects (name, age, salary) using lambda expressions.

Code:-

```
import java.util.*;
class Employee {
    String name;
    int age;
    double salary;
    public Employee(String name, int age, double salary) {
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
    @Override
    public String toString() {
        return name + " - Age: " + age + ", Salary: " + salary;
    }
}
public class Main {
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Rishu", 20, 50000));
```

```
        employees.add(new Employee("Shalini", 21, 60000));
        employees.add(new Employee("Shreya", 24, 55000));
        employees.sort(Comparator.comparing(emp -> emp.name));
        System.out.println("Sorted by Name:");
        employees.forEach(System.out::println);
        employees.sort(Comparator.comparingInt(emp -> emp.age));
        System.out.println("\nSorted by Age:");
        employees.forEach(System.out::println);
        employees.sort(Comparator.comparingDouble(emp -> emp.salary));
        System.out.println("\nSorted by Salary:");
        employees.forEach(System.out::println);
    }
}
```

OUTPUT:-



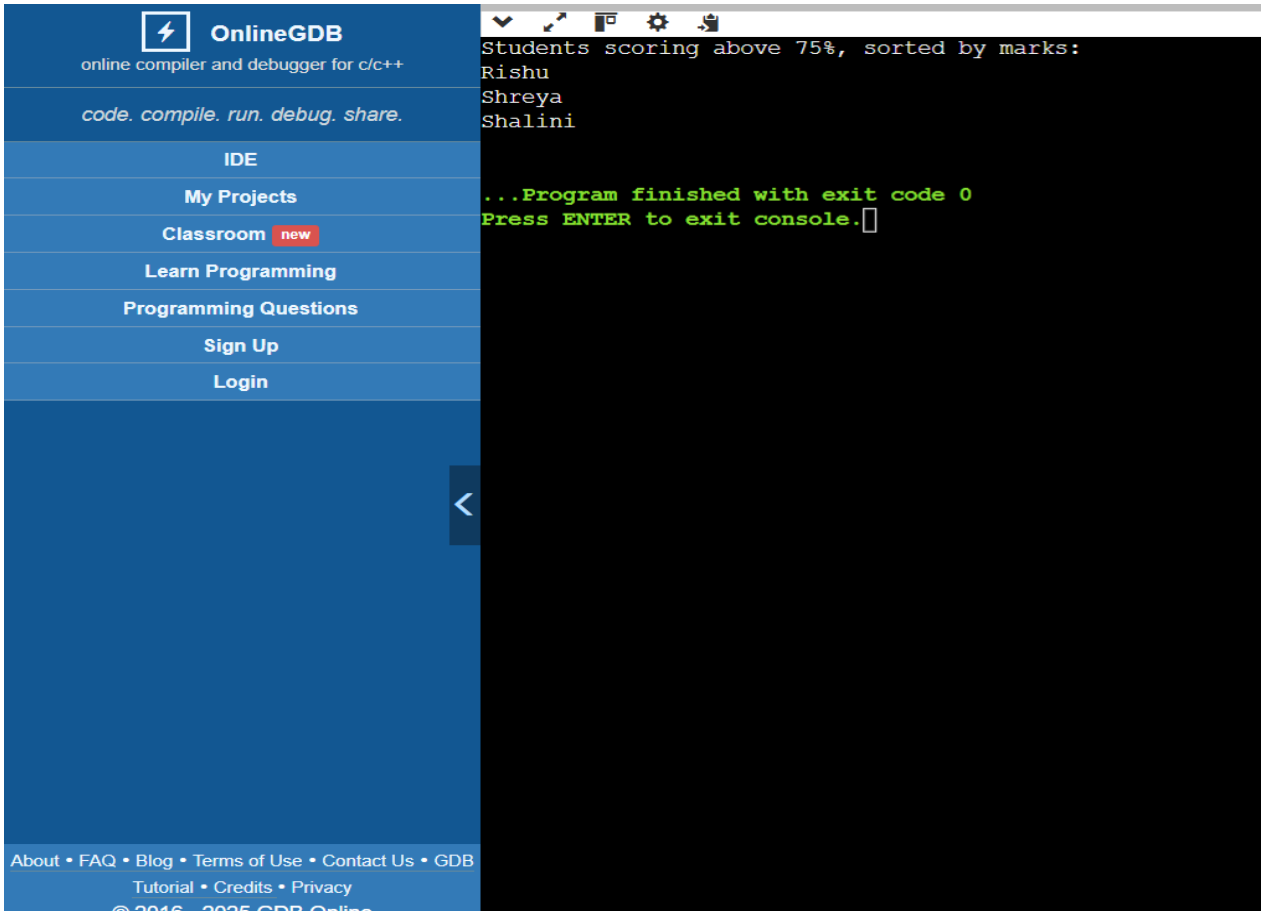
Ques 2:-

Create a program to use lambda expressions and stream operations to filter students scoring above 75%, sort them by marks, and display their names.

Code:-

```
import java.util.*;

import java.util.stream.Collectors;

class Student {

    String name;

    double marks;

    public Student(String name, double marks) {

        this.name = name;

        this.marks = marks;

    }}

public class Main {

    public static void main(String[] args) {

        List<Student> students = new ArrayList<>();

        students.add(new Student("Rishu", 80.5));

        students.add(new Student("Samir", 72.0));

        students.add(new Student("Shalini", 90.0));

        students.add(new Student("Dramveer", 60.5));

        students.add(new Student("Shreya", 85.0));

        List<String> topStudents = students.stream()

                .filter(s -> s.marks > 75)

                .sorted(Comparator.comparingDouble(s -> s.marks))

                .map(s -> s.name)

                .collect(Collectors.toList());

        System.out.println("Students scoring above 75%, sorted by marks:");

        topStudents.forEach(System.out::println);

    }}
```

OUTPUT:-



LEARNING OUTCOMES:-

- **Mastering Lambda Expressions** – Learn how to use lambda expressions for sorting and filtering collections efficiently.
- **Using Java Streams API** – Understand how to filter, sort, and transform data using stream operations like `filter()`, `sorted()`, and `map()`.
- **Comparator and Sorting Techniques** – Gain hands-on experience with `Comparator.comparing()` and how to sort objects dynamically based on different attributes.
- **Functional Programming in Java** – Improve your ability to write clean, concise, and efficient code using functional programming concepts.
- **Efficient Data Processing** – Learn how to manipulate collections using `forEach()`, `Collectors.toList()`, and other stream functions for optimized performance.