Ques1)

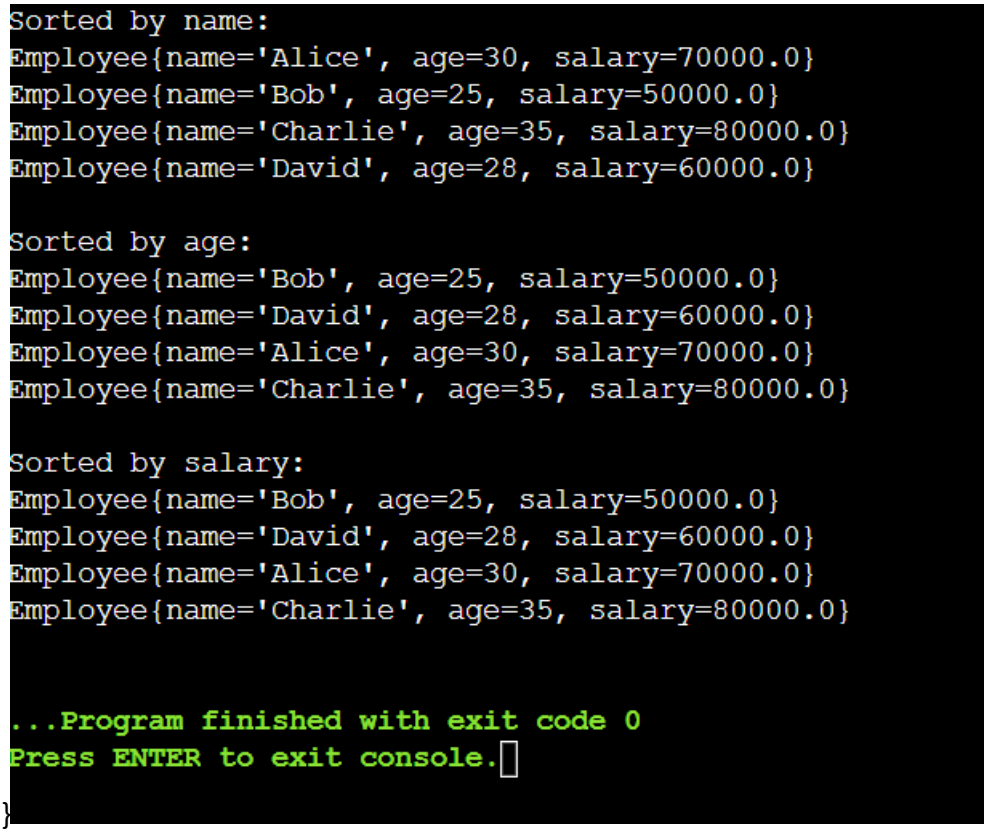
Write a program to sort a list of Employee objects (name, age, salary) using lambda expressions.

Code

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

class Employee {
	private String name;
	private int age;
	private double salary;

	public Employee(String name, int age, double salary) {
		this.name = name;
		this.age = age;
		this.salary = salary;
	}

	public String getName() {
		return name;
	}

	public int getAge() {
		return age;
	}
```

```
    public double getSalary() {
        return salary;
    }

    @Override
    public String toString() {
        return "Employee{name='" + name + "', age=" + age + ", salary=" + salary + "}";
    }
}

public class Main {    // Changed from EmployeeSorter to Main
    public static void main(String[] args) {
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Alice", 30, 70000));
        employees.add(new Employee("Bob", 25, 50000));
        employees.add(new Employee("Charlie", 35, 80000));
        employees.add(new Employee("David", 28, 60000));

        // Sorting by name
        employees.sort(Comparator.comparing(Employee::getName));
        System.out.println("Sorted by name:");
        employees.forEach(System.out::println);

        // Sorting by age
        employees.sort(Comparator.comparingInt(Employee::getAge));
```

```
        System.out.println("\nSorted by age:");
        employees.forEach(System.out::println);

        // Sorting by salary
        employees.sort(Comparator.comparingDouble(Employee::getSalary));
        System.out.println("\nSorted by salary:");
        employees.forEach(System.out::println);
    }
```




```
}
```

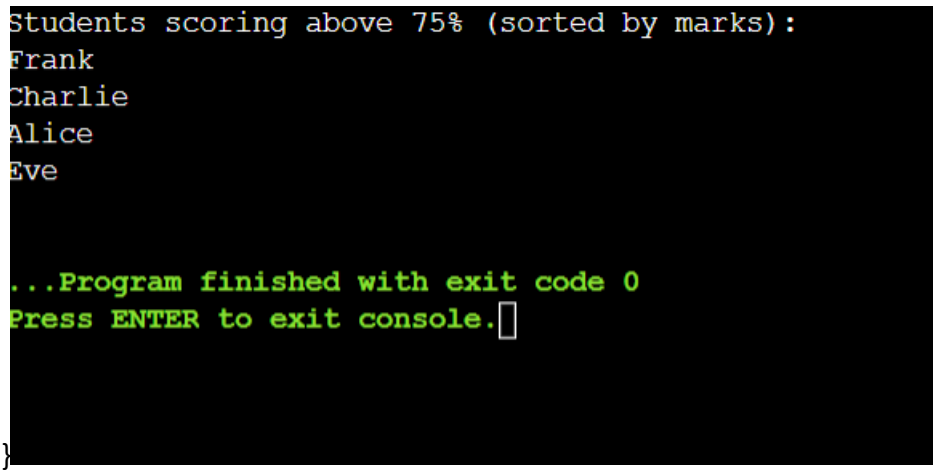
Ques 2)

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.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.stream.Collectors;

class Student {
	String name;
	double marks;

	Student(String name, double marks) {
		this.name = name;
		this.marks = marks;
	}

	public String getName() {
		return name;
	}

	public double getMarks() {
		return marks;
	}
}

public class Main {
	public static void main(String[] args) {
```

```
        List<Student> students = new ArrayList<>();
        students.add(new Student("Alice", 85.0));
        students.add(new Student("Bob", 70.0));
        students.add(new Student("Charlie", 90.0));
        students.add(new Student("David", 60.0));
        students.add(new Student("Eve", 78.0));
        students.add(new Student("Frank", 92.0));

        List<String> filteredAndSortedStudentNames = students.stream()
                .filter(student -> student.getMarks() > 75)
                .sorted(Comparator.comparingDouble(Student::getMarks).reversed())
                .map(Student::getName)
                .collect(Collectors.toList());

        System.out.println("Students scoring above 75% (sorted by marks):");
        filteredAndSortedStudentNames.forEach(System.out::println);
    }
```




}

Ques 3)
Write a Java program to process a large dataset of products using streams. Perform operations such as grouping products by category, finding the most expensive product in each category, and calculating the average price of all products.

Code

```
import java.util.*;

import java.util.stream.Collectors;


class Product {

	private String name;

	private String category;

	private double price;


	public Product(String name, String category, double price) {

		this.name = name;

		this.category = category;

		this.price = price;

	}


	public String getName() {

		return name;

	}


	public String getCategory() {

		return category;

	}
```

```
    public double getPrice() {
        return price;
    }

    @Override
    public String toString() {
        return String.format("Product{name='%s', category='%s', price=%.2f}", name,
category, price);
    }
}

public class Main {
    public static void main(String[] args) {
        // Create a list of products
        List<Product> products = Arrays.asList(
            new Product("Laptop", "Electronics", 1200.00),
            new Product("Smartphone", "Electronics", 800.00),
            new Product("Tablet", "Electronics", 300.00),
            new Product("Chair", "Furniture", 150.00),
            new Product("Table", "Furniture", 250.00),
            new Product("Sofa", "Furniture", 700.00),
            new Product("Shirt", "Clothing", 50.00),
            new Product("Jeans", "Clothing", 80.00),
            new Product("Jacket", "Clothing", 120.00)
        );
```

```
        Map<String, List<Product>> productsByCategory = products.stream()
                .collect(Collectors.groupingBy(Product::getCategory));

        Map<String, Product> mostExpensiveProducts =
productsByCategory.entrySet().stream()
                .collect(Collectors.toMap(
                        Map.Entry::getKey,
                        entry -> entry.getValue().stream()
                                .max(Comparator.comparingDouble(Product::getPrice))
                                .orElse(null)
                ));

        double averagePrice = products.stream()
                .collect(Collectors.averagingDouble(Product::getPrice));

        System.out.println("Most Expensive Products by Category:");
        mostExpensiveProducts.forEach((category, product) -> {
                System.out.println(category + ": " + product);
        });

        System.out.printf("Average Price of All Products: %.2f%n", averagePrice);
    }
```

}

```
Most Expensive Products by Category:
Clothing: Product{name='Jacket', category='Clothing', price=120.00}
Electronics: Product{name='Laptop', category='Electronics', price=1200.00}
Furniture: Product{name='Sofa', category='Furniture', price=700.00}
Average Price of All Products: 405.56


...Program finished with exit code 0
Press ENTER to exit console.
```