NAME:	BIPASHA
UID:	23BCS10735
SECTION:	622-A
SUBJECT:	JAVA

Practice -2

(A)CODE:

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

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 lamba1 {
    public static void main(String[] args) {
```

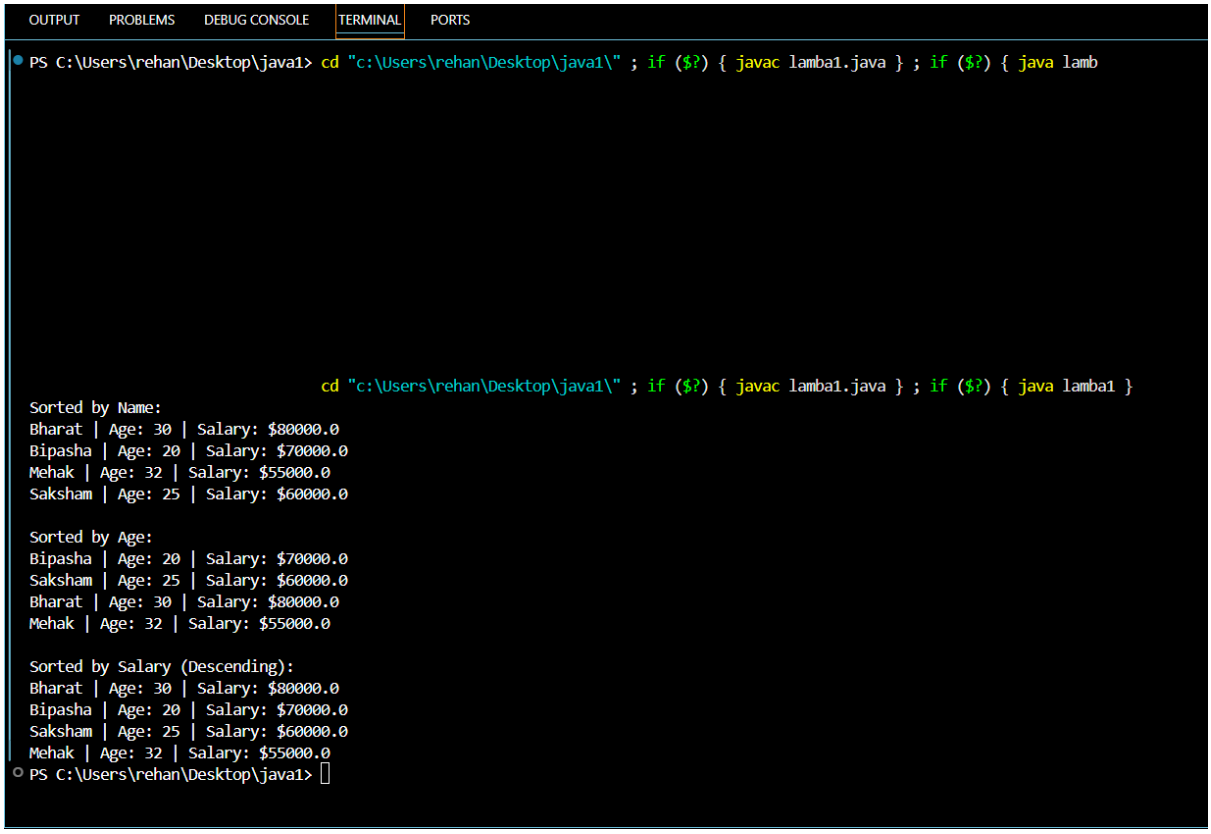
```
        List<Employee> employees = new ArrayList<>();
        employees.add(new Employee("Bipasha", 20, 70000));
        employees.add(new Employee("Mehak", 32, 55000));
        employees.add(new Employee("Saksham", 25, 60000));
        employees.add(new Employee("Bharat", 30, 80000));

        employees.sort((e1, e2) -> e1.name.compareTo(e2.name));
        System.out.println("Sorted by Name:");
        employees.forEach(System.out::println);

        employees.sort((e1, e2) -> Integer.compare(e1.age, e2.age));
        System.out.println("\nSorted by Age:");
        employees.forEach(System.out::println);

        employees.sort((e1, e2) -> Double.compare(e2.salary, e1.salary));
        System.out.println("\nSorted by Salary (Descending):");
        employees.forEach(System.out::println);
    }
}
```

OUTPUT:l



(B)

Code:

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

class Student {
    String name;
    double marks;

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

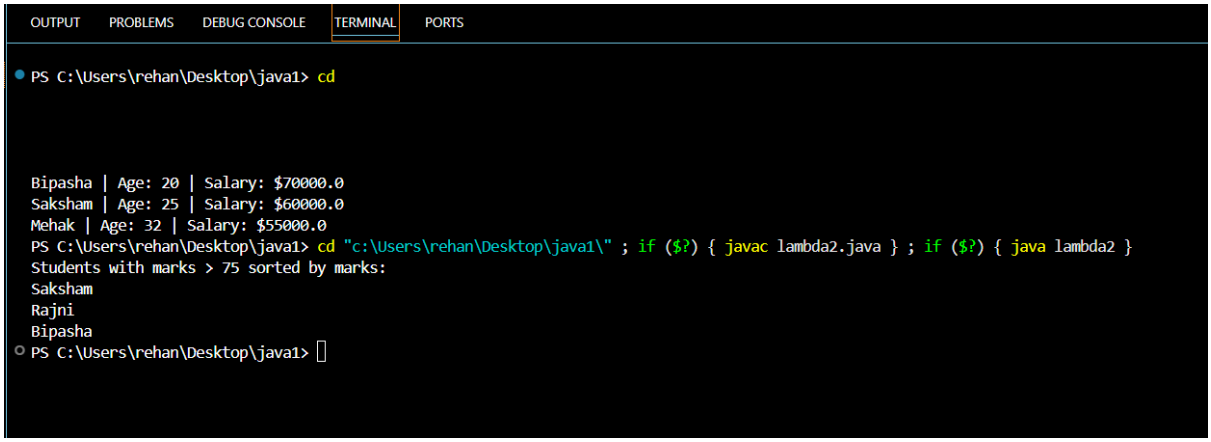
```
    }
}

public class lambda2 {
    public static void main(String[] args) {
        List<Student> students = new ArrayList<>();
        students.add(new Student("Bipasha", 100));
        students.add(new Student("Bharat", 68));
        students.add(new Student("Rajni", 91));
        students.add(new Student("Mehak", 74));
        students.add(new Student("Saksham", 88));

        List<String> topStudents = students.stream()
                .filter(s -> s.marks > 75)
                .sorted((s1, s2) -> Double.compare(s1.marks, s2.marks))
                .map(s -> s.name)
                .collect(Collectors.toList());

        System.out.println("Students with marks > 75 sorted by marks:");
        topStudents.forEach(System.out::println);
    }
}
```

Output:



(c)

Code:

```
import java.util.*;
import java.util.stream.Collectors;

class Product {
    String name;
    double price;
    String category;

    public Product(String name, double price, String category) {
        this.name = name;
        this.price = price;
        this.category = category;
    }

    @Override
    public String toString() {
        return name + " | $" + price;
    }
```

```
}

public class lamda3 {
    public static void main(String[] args) {
        List<Product> products = Arrays.asList(
                new Product("Laptop", 1200, "Electronics"),
                new Product("Smartphone", 800, "Electronics"),
                new Product("Headphones", 150, "Electronics"),
                new Product("Shirt", 50, "Clothing"),
                new Product("Jeans", 70, "Clothing"),
                new Product("Jacket", 120, "Clothing"),
                new Product("Coffee Maker", 90, "Home Appliances"),
                new Product("Blender", 60, "Home Appliances")
        );

        Map<String, List<Product>> productsByCategory = products.stream()
                .collect(Collectors.groupingBy(p -> p.category));

        System.out.println("Products Grouped by Category:");
        productsByCategory.forEach((category, list) ->
System.out.println(category + ": " + list));

        Map<String, Optional<Product>> maxPriceByCategory =
products.stream()
                .collect(Collectors.groupingBy(p -> p.category,
                        Collectors.maxBy(Comparator.comparingDouble(p ->
p.price))));
```

```
        System.out.println("\nMost Expensive Product in Each Category:");
        maxPriceByCategory.forEach((category, product) -> product.ifPresent(p -> System.out.println(category + ": " + p)));


        double avgPrice = products.stream()
                .collect(Collectors.averagingDouble(p -> p.price));
        System.out.println("\nAverage Price of All Products: $" + avgPrice);
    }
}
```

Output: