# DATA STRUCTURES AND ALGORITHMS

**Superset ID :** 6384831

**Name :** Mohana Priya N

**E-mail :** mohanapriya.2205056@srec.ac.in

## Mandatory Questions:

**1) Exercise 2: E-commerce Platform Search Function**

**Solution:**

**//ProductSearch.java**

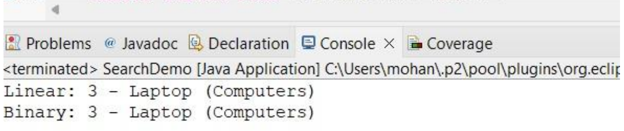
```
package palindrome;
public class ProductSearch {
    int productId;
    String productName;
    String category;
    public ProductSearch(int id, String name, String cat) {
        this.productId = id;
        this.productName = name;
        this.category = cat;
    }
    public String toString() {
        return productId + " - " + productName + " (" + category + ")";
    }
}
```

**//SearchDemo.java**

```
package palindrome;
import java.util.Arrays;
import java.util.Comparator;
public class SearchDemo {
    public static ProductSearch linearSearch(ProductSearch[] arr, String name) {
```

```
for (ProductSearch p : arr) {
            if (p.productName.equalsIgnoreCase(name)) return p;
        }
        return null;
    }
    public static ProductSearch binarySearch(ProductSearch[] arr, String name) {
        int low = 0, high = arr.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            int cmp = arr[mid].productName.compareToIgnoreCase(name);
            if (cmp == 0) return arr[mid];
            else if (cmp < 0) low = mid + 1;
            else high = mid - 1; }
        return null; }
    public static void main(String[] args) {
        ProductSearch[] items = {
            new ProductSearch(1, "Book", "Education"),
            new ProductSearch(2, "Charger", "Electronics"),
            new ProductSearch(3, "Laptop", "Computers")
        };
        Arrays.sort(items, Comparator.comparing(p -> p.productName));
        System.out.println("Linear: " + linearSearch(items, "Laptop"));
        System.out.println("Binary: " + binarySearch(items, "Laptop"));
    }}
```

**Output:**



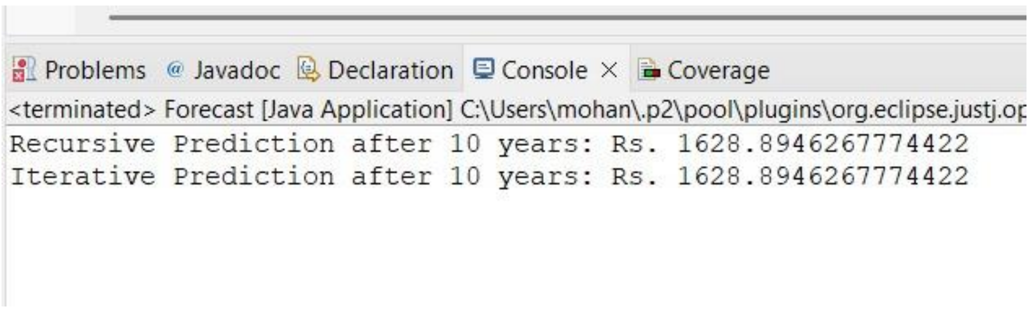
```
<terminated> SearchDemo [Java Application] C:\Users\mohan\.p2\pool\plugins\org.eclip
Linear: 3 - Laptop (Computers)
Binary: 3 - Laptop (Computers)
```

**2) Exercise 7: Financial Forecasting**

**Solution:**

```
package palindrome;

public class Forecast {

    public static double predictRecursive(double currentValue, double rate, int years) {

        if (years == 0) return currentValue;

        return predictRecursive(currentValue * (1 + rate), rate, years - 1);

    }

    public static double predictIterative(double currentValue, double rate, int years) {

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

            currentValue *= (1 + rate);

        }

        return currentValue;

    }

    public static void main(String[] args) {

        double start = 1000;

        double rate = 0.05;

        int years = 10;

        System.out.println("Recursive Prediction after " + years + " years: Rs. " +
predictRecursive(start, rate, years));

        System.out.println("Iterative Prediction after " + years + " years: Rs. " +
predictIterative(start, rate, years));

    }}
```

**Output:**



```
<terminated> Forecast [Java Application] C:\Users\mohan\.p2\pool\plugins\org.eclipse.justj.op
Recursive Prediction after 10 years: Rs. 1628.8946267774422
Iterative Prediction after 10 years: Rs. 1628.8946267774422
```

**Other Questions:**

**3) Exercise 1: Inventory Management System**

**Solution:**

**//Product.java**

```
package palindrome;
public class Product {
    int productId;
    String productName;
    int quantity;
    double price;
    public Product(int id, String name, int qty, double price) {
        this.productId = id;
        this.productName = name;
        this.quantity = qty;
        this.price = price;
    }
}
```

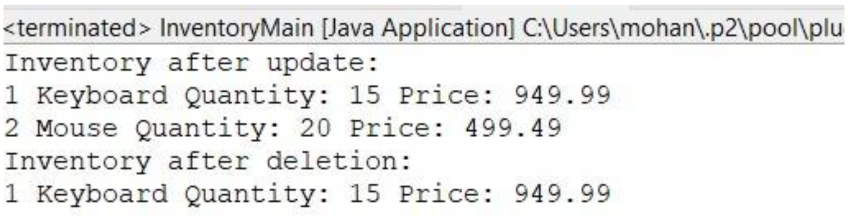
**//Inventory.java**

```
package palindrome;
import java.util.*;
public class Inventory {
    HashMap<Integer, Product> products = new HashMap<>();
    public void addProduct(Product p) {
        products.put(p.productId, p);
    }
    public void updateProduct(int id, int quantity, double price) {
        if (products.containsKey(id)) {
            Product p = products.get(id);
            p.quantity = quantity;
            p.price = price;
        }
    }
    public void deleteProduct(int id) {
        products.remove(id);
    }
    public void displayAll() {
        for (Product p : products.values()) {
            System.out.println(p.productId + " " + p.productName + " Quantity: " + p.quantity + "
Price: " + p.price);
        }
    }
}
```

**//InventoryMain.java**

```
package palindrome;
public class InventoryMain {
    public static void main(String[] args) {
        Inventory inv = new Inventory();
        inv.addProduct(new Product(1, "Keyboard", 10, 999.99));
        inv.addProduct(new Product(2, "Mouse", 20, 499.49));
        inv.updateProduct(1, 15, 949.99);
        System.out.println("Inventory after update:");
        inv.displayAll();
        inv.deleteProduct(2);
        System.out.println("Inventory after deletion:");
        inv.displayAll();
    }
}
```

**Output:**



```
<terminated> InventoryMain [Java Application] C:\Users\mohan\.p2\pool\plu
Inventory after update:
1 Keyboard Quantity: 15 Price: 949.99
2 Mouse Quantity: 20 Price: 499.49
Inventory after deletion:
1 Keyboard Quantity: 15 Price: 949.99
```

**4) Exercise 3: Sorting Customer Orders**

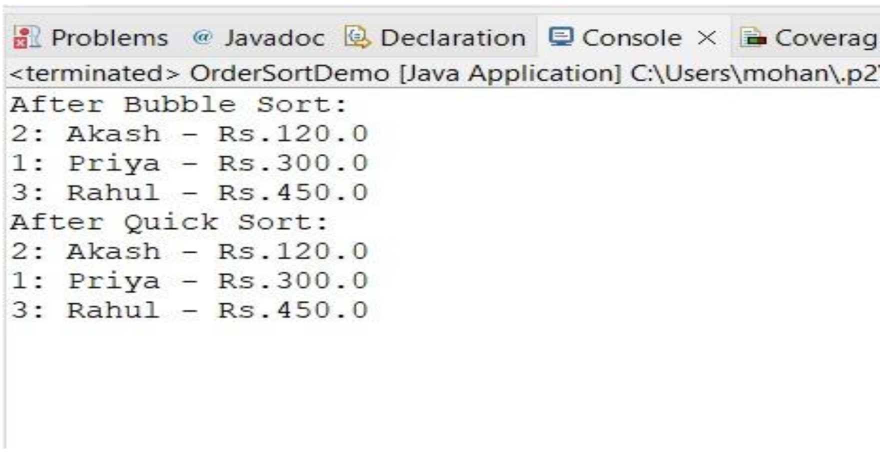
**Solution:**

**//OrderSortDemo.java**

```
package palindrome;
public class OrderSortDemo {
    static class Order {
        int orderId;
        String customerName;
        double totalPrice;
        public Order(int id, String name, double price) {
            this.orderId = id;
            this.customerName = name;
            this.totalPrice = price;
        }
        @Override
        public String toString() {
            return orderId + ": " + customerName + " - Rs." + totalPrice;
        }
    }
    static class OrderSorter {
        public static void bubbleSort(Order[] arr) {
            int n = arr.length;
            for (int i = 0; i < n - 1; i++)
                for (int j = 0; j < n - i - 1; j++)
                    if (arr[j].totalPrice > arr[j + 1].totalPrice) {
                        Order temp = arr[j];
                        arr[j] = arr[j + 1];
                        arr[j + 1] = temp;
                    }
        }
        public static void quickSort(Order[] arr, int low, int high) {
            if (low < high) {
                int pi = partition(arr, low, high);
                quickSort(arr, low, pi - 1);
                quickSort(arr, pi + 1, high);
            }
        }
        private static int partition(Order[] arr, int low, int high) {
            double pivot = arr[high].totalPrice;
            int i = low - 1;
            for (int j = low; j < high; j++) {
                if (arr[j].totalPrice <= pivot) {
                    i++;
                    Order temp = arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
            Order temp = arr[i + 1];
            arr[i + 1] = arr[high];
            arr[high] = temp;
            return i + 1;
        }}
```

```
    public static void main(String[] args) {
        Order[] orders = {
            new Order(1, "Priya", 300),
            new Order(2, "Akash", 120),
            new Order(3, "Rahul", 450)
        };
        OrderSorter.bubbleSort(orders);
        System.out.println("After Bubble Sort:");
        for (Order o : orders)
            System.out.println(o);
        OrderSorter.quickSort(orders, 0, orders.length - 1);
        System.out.println("After Quick Sort:");
        for (Order o : orders)
            System.out.println(o);
    }
}
```

**Output:**



```
<terminated> OrderSortDemo [Java Application] C:\Users\mohan\.p2
After Bubble Sort:
2: Akash - Rs.120.0
1: Priya - Rs.300.0
3: Rahul - Rs.450.0
After Quick Sort:
2: Akash - Rs.120.0
1: Priya - Rs.300.0
3: Rahul - Rs.450.0
```

**5) Exercise 4: Employee Management System**

**Solution:**

**//Employee.java**

```
package palindrome;
public class Employee {
    int employeeId;
    String name;
    String position;
    double salary;
    public Employee(int id, String name, String position, double salary) {
        this.employeeId = id;
        this.name = name;
        this.position = position;
        this.salary = salary;
    }
}
```

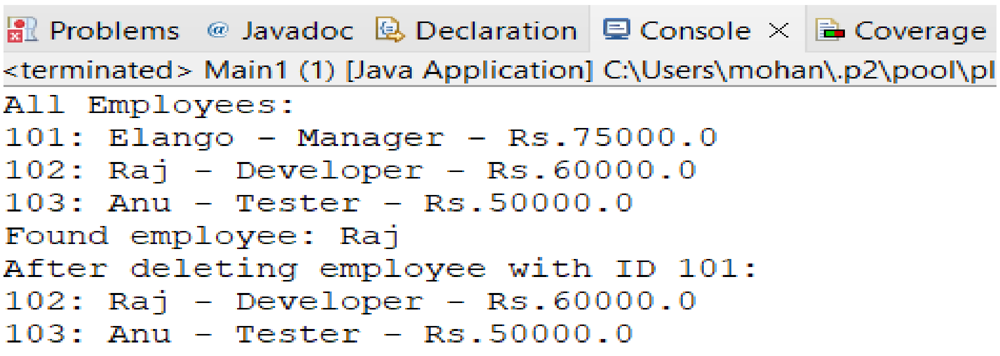
**//EmployeeSystem.java**

```
package palindrome;
public class EmployeeSystem {
    Employee[] employees = new Employee[100];
    int count = 0;
    public void addEmployee(Employee emp) {
        employees[count++] = emp;
    }
    public Employee searchEmployee(int id) {
        for (int i = 0; i < count; i++) {
            if (employees[i].employeeId == id) return employees[i];
        }
        return null;
    }
    public void deleteEmployee(int id) {
        for (int i = 0; i < count; i++) {
            if (employees[i].employeeId == id) {
                for (int j = i; j < count - 1; j++) {
                    employees[j] = employees[j + 1];
                }
                count--;
                break;
            }
        }}
    public void displayAll() {
        for (int i = 0; i < count; i++) {
            Employee e = employees[i];
            System.out.println(e.employeeId + ": " + e.name + " - " + e.position + " - Rs." +
e.salary);
        }
    }
}
```

**//Main1.java**

```
package palindrome;
public class Main1 {
                public static void main(String[] args) {
                    EmployeeSystem empSys = new EmployeeSystem();
                    empSys.addEmployee(new Employee(101, "Elango", "Manager", 75000));
                    empSys.addEmployee(new Employee(102, "Raj", "Developer", 60000));
                    empSys.addEmployee(new Employee(103, "Anu", "Tester", 50000));
                    System.out.println("All Employees:");
                    empSys.displayAll();
                    Employee emp = empSys.searchEmployee(102);
                    if (emp != null) {
                        System.out.println("Found employee: " + emp.name);
                    } else {
                        System.out.println("Employee not found");
                    }
                    empSys.deleteEmployee(101);
                    System.out.println("After deleting employee with ID 101:");
                    empSys.displayAll();
                }
            }
```

**Output:**



```
<terminated> Main1 (1) [Java Application] C:\Users\mohan\.p2\pool\pl
All Employees:
101: Elango - Manager - Rs.75000.0
102: Raj - Developer - Rs.60000.0
103: Anu - Tester - Rs.50000.0
Found employee: Raj
After deleting employee with ID 101:
102: Raj - Developer - Rs.60000.0
103: Anu - Tester - Rs.50000.0
```

**6) Exercise 5: Task Management System**

**Solution:**

**//Task.java**

```
package palindrome;
public class Task {
    int taskId;
    String taskName;
    String status;
    Task next;
    public Task(int id, String name, String status) {
        this.taskId = id;
        this.taskName = name;
        this.status = status;
        this.next = null;
    }
}
```

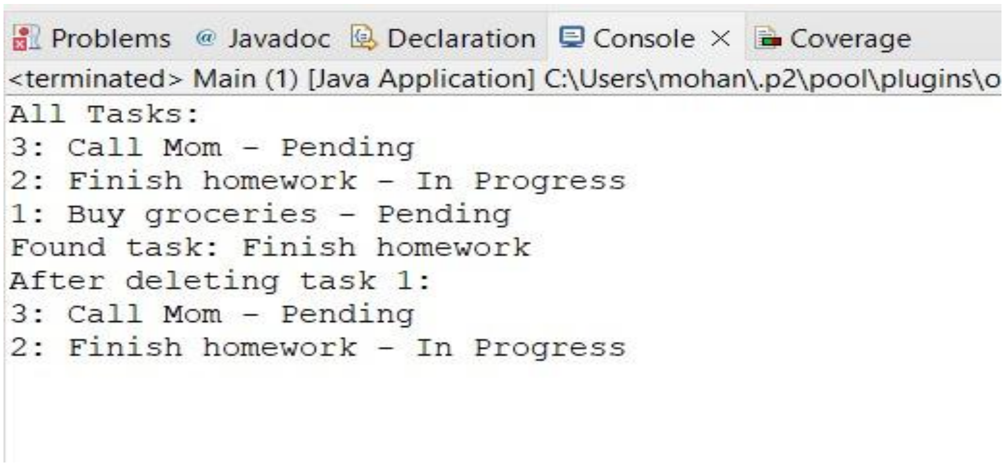
**//TaskList.java**

```
package palindrome;
public class TaskList {
    Task head;
    public void addTask(Task newTask) {
        newTask.next = head;
        head = newTask;
    }
    public Task searchTask(int id) {
        Task temp = head;
        while (temp != null) {
            if (temp.taskId == id) return temp;
            temp = temp.next;
        }
        return null;
    }
    public void deleteTask(int id) {
        if (head == null) return;
        if (head.taskId == id) {
            head = head.next;
            return;
        }
        Task current = head;
        while (current.next != null) {
            if (current.next.taskId == id) {
                current.next = current.next.next;
                return;
            }
            current = current.next;
        }
    }
    public void displayTasks() {
        Task temp = head;
        while (temp != null) {
            System.out.println(temp.taskId + ": " + temp.taskName + " - " + temp.status);
            temp = temp.next;
        }
    }
}
```

**//Main.java**

```
package palindrome;
public class Main {
    public static void main(String[] args) {
        TaskList taskList = new TaskList();
        taskList.addTask(new Task(1, "Buy groceries", "Pending"));
        taskList.addTask(new Task(2, "Finish homework", "In Progress"));
        taskList.addTask(new Task(3, "Call Mom", "Pending"));
        System.out.println("All Tasks:");
        taskList.displayTasks();
        Task task = taskList.searchTask(2);
        if (task != null) {
            System.out.println("Found task: " + task.taskName);
        } else {
            System.out.println("Task not found");
        }
        taskList.deleteTask(1);
        System.out.println("After deleting task 1:");
        taskList.displayTasks();
    }}
```

**Output:**



```
<terminated> Main (1) [Java Application] C:\Users\mohan\.p2\pool\plugins\o
All Tasks:
3: Call Mom - Pending
2: Finish homework - In Progress
1: Buy groceries - Pending
Found task: Finish homework
After deleting task 1:
3: Call Mom - Pending
2: Finish homework - In Progress
```

**7) Exercise 6: Library Management System**

**Solution:**

**//Book.java**

```
package palindrome;
public class Book {
    public int bookId;
    public String title;
    public String author;
    public Book(int id, String title, String author) {
        this.bookId = id;
        this.title = title;
        this.author = author;
    }
}
```

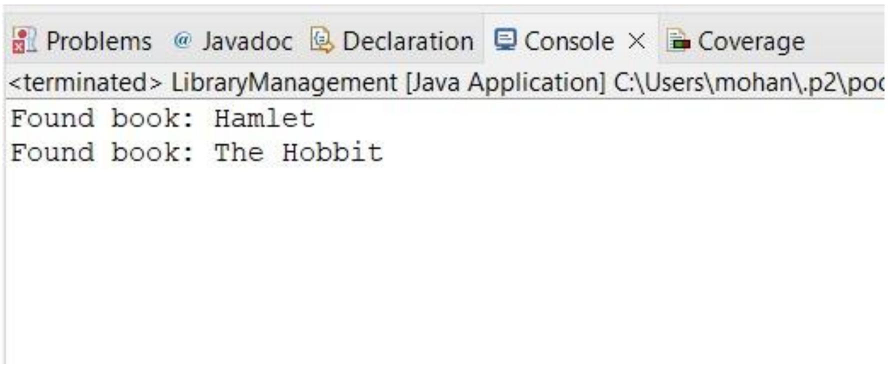
**//LibrarySearch.java**

```
package palindrome;
import java.util.Arrays;
import java.util.Comparator;
public class LibrarySearch {
    public static Book linearSearch(Book[] books, String title) {
        for (Book b : books) {
            if (b.title.equalsIgnoreCase(title))
                return b;
        }
        return null;
    }
    public static Book binarySearch(Book[] books, String title) {
        int low = 0, high = books.length - 1;
        while (low <= high) {
            int mid = (low + high) / 2;
            int cmp = books[mid].title.compareToIgnoreCase(title);
            if (cmp == 0)
                return books[mid];
            else if (cmp < 0)
                low = mid + 1;
            else
                high = mid - 1;
        }
        return null;
    }
}
```

**//LibraryManagement.java**

```
package palindrome;
import java.util.Arrays;
import java.util.Comparator;
public class LibraryManagement {
    public static void main(String[] args) {
        Book[] books = {
            new Book(1, "The Hobbit", "J.R.R. Tolkien"),
            new Book(2, "1984", "George Orwell"),
            new Book(3, "Hamlet", "William Shakespeare")
        };
        Arrays.sort(books, Comparator.comparing(b -> b.title));
        Book found = LibrarySearch.linearSearch(books, "Hamlet");
        if (found != null)
            System.out.println("Found book: " + found.title);
        else
            System.out.println("Book not found");
        Book foundBinary = LibrarySearch.binarySearch(books, "The Hobbit");
        if (foundBinary != null)
            System.out.println("Found book: " + foundBinary.title);
        else
            System.out.println("Book not found");
    }
}
```

**Output:**



```
<terminated> LibraryManagement [Java Application] C:\Users\mohan\.p2\poo
Found book: Hamlet
Found book: The Hobbit
```