Data structures and Algorithms Week-1

Exercise 1: E-commerce Platform Search Function

Scenario:

You are working on the search functionality of an e-commerce platform. The search needs to be optimized for fast performance.

Steps:

1. **Understand Asymptotic Notation:**
    - Explain Big O notation and how it helps in analyzing algorithms.
    - Describe the best, average, and worst-case scenarios for search operations.
2. **Setup:**
    - Create a class **Product** with attributes for searching, such as **productId, productName**, and **category**.
3. **Implementation:**
    - Implement linear search and binary search algorithms.
    - Store products in an array for linear search and a sorted array for binary search.
4. **Analysis:**
    - Compare the time complexity of linear and binary search algorithms.
    - Discuss which algorithm is more suitable for your platform and why.

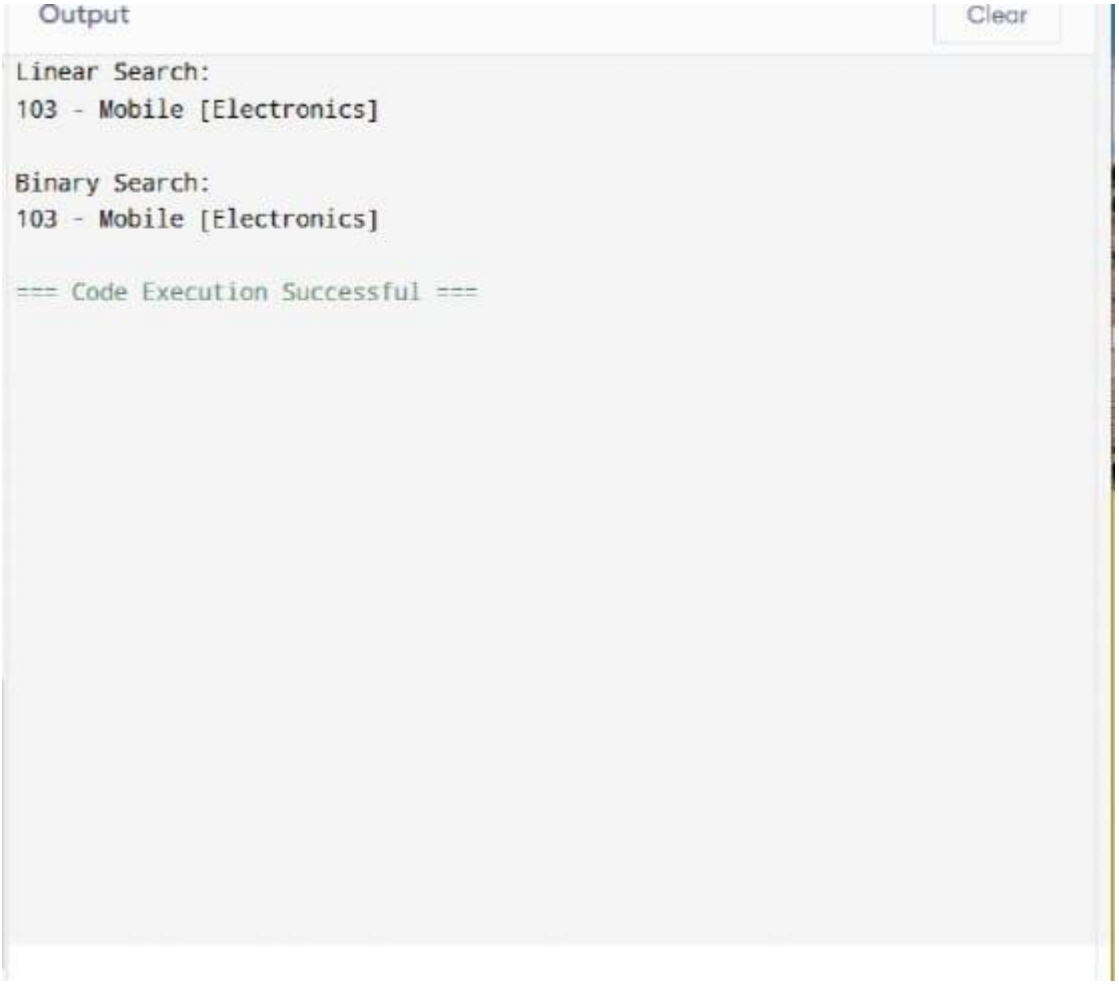
Program:

```
import java.util.Arrays;
import java.util.Comparator;
public class EcommerceSearch {
   static class Product {
      int productId;
      String productName;
      String category;
public Product(int productId, String productName, String category) {
         this.productId = productId;
         this.productName = productName;
         this.category = category;
```

```
    }
public String toString() {
        return productId + " - " + productName + " [" + category + "]";
    }
  }
  public static Product linearSearch(Product[] products, String targetName) {
    for (Product product : products) {
      if (product.productName.equalsIgnoreCase(targetName)) {
        return product;
      }
    }
    return null;
  }
  public static Product binarySearch(Product[] products, String targetName) {
    int left = 0, right = products.length - 1;
 while (left <= right) {
      int mid = (left + right) / 2;
      int cmp = products[mid].productName.compareToIgnoreCase(targetName);
if (cmp == 0) return products[mid];
      else if (cmp < 0) left = mid + 1;
      else right = mid - 1;
    }
return null;
  }
  public static void main(String[] args) {
    Product[] products = {
      new Product(101, "Laptop", "Electronics"),
      new Product(102, "Shoes", "Fashion"),
      new Product(103, "Mobile", "Electronics"),
      new Product(104, "Book", "Stationery"),
      new Product(105, "Watch", "Accessories")
```

```
        };
        System.out.println("Linear Search:");
        Product found1 = linearSearch(products, "Mobile");
        System.out.println(found1 != null ? found1 : "Product not found");
        Arrays.sort(products, Comparator.comparing(p -> p.productName.toLowerCase()));
        System.out.println("\nBinary Search:");
        Product found2 = binarySearch(products, "Mobile");
        System.out.println(found2 != null ? found2 : "Product not found");
    }
}
```

Out put:



Data structures and Algorithms Week-1

Exercise 2: Financial Forecasting

Scenario:

You are developing a financial forecasting tool that predicts future values based on past data.

Steps:

1. **Understand Recursive Algorithms:**
   - Explain the concept of recursion and how it can simplify certain problems.
2. **Setup:**
   - Create a method to calculate the future value using a recursive approach.
3. **Implementation:**
   - Implement a recursive algorithm to predict future values based on past growth rates.
4. **Analysis:**
   - Discuss the time complexity of your recursive algorithm.
   - Explain how to optimize the recursive solution to avoid excessive computation.

Program:

```
public class FinancialForecasting {
 public static double predictFutureValue(double currentValue, double growthRate, int years) {
       if (years == 0) {
          return currentValue;
       }
return predictFutureValue(currentValue * (1 + growthRate), growthRate, years - 1);
   }
 public static void main(String[] args) {
       double initialValue = 10000;
       double growthRate = 0.08;
       int years = 5;
double futureValue = predictFutureValue(initialValue, growthRate, years);
       System.out.printf("Predicted value after %d years: ₹%.2f\n", years, futureValue);
   }
}
```

-6427402

Out put:


Output
Clear
Predicted value after 5 years: ?14693.28
=== Code Execution Successful ===