WEEK 1 DATA STRUCTURE AND ALGORITHAM

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.

CODE:

PRODUCT.JAVA

```
package search;

public 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 "Product ID: " + productId + ", Name: " + productName + ", Category: " +
category;

          }

}
```

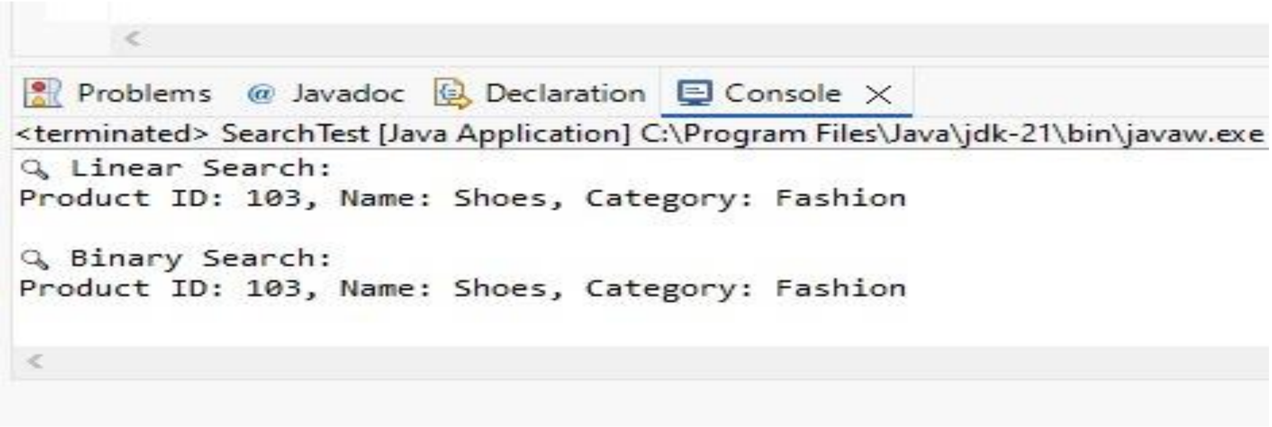
PRODUCTSEARCH.JAVA

```
package search;

import java.util.Arrays;

import java.util.Comparator;

public class ProductSearch {

         public static Product linearSearch(Product[] products, int productId) {

     for (Product p : products) {

        if (p.productId == productId) {

           return p;

        }
        }
        return null;
    }
    public static Product binarySearch(Product[] products, int productId) {
        int low = 0, high = products.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (products[mid].productId == productId) {
                return products[mid];
            } else if (products[mid].productId < productId) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return null;
    }
    public static void sortByProductId(Product[] products) {
        Arrays.sort(products, Comparator.comparingInt(p -> p.productId));
    }

}
```

SEARCHTEST.JAVA

```
package search;
public class SearchTest {
              public static void main(String[] args) {
        Product[] products = {
            new Product(104, "Mouse", "Electronics"),
            new Product(101, "Laptop", "Electronics"),
            new Product(103, "Shoes", "Fashion"),
            new Product(102, "Watch", "Fashion")
        };
        int searchId = 103;
        System.out.println("🔍 Linear Search:");
        Product result1 = ProductSearch.linearSearch(products, searchId);
        System.out.println(result1 != null ? result1 : "Product not found");
        ProductSearch.sortByProductId(products);
        System.out.println("\n🔍 Binary Search:");
        Product result2 = ProductSearch.binarySearch(products, searchId);
        System.out.println(result2 != null ? result2 : "Product not found");
    }

}
```

OUTPUT



```
<terminated> SearchTest [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe
🔍 Linear Search:
Product ID: 103, Name: Shoes, Category: Fashion

🔍 Binary Search:
Product ID: 103, Name: Shoes, Category: Fashion
```

2: Financial Forecasting

Scenario:

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

CODE:

FINANCIALFORECAST.JAVA

```
package FF;
public class FinancialForecast {
        public static double futureValue(double initialValue, double growthRate, int years) {
```

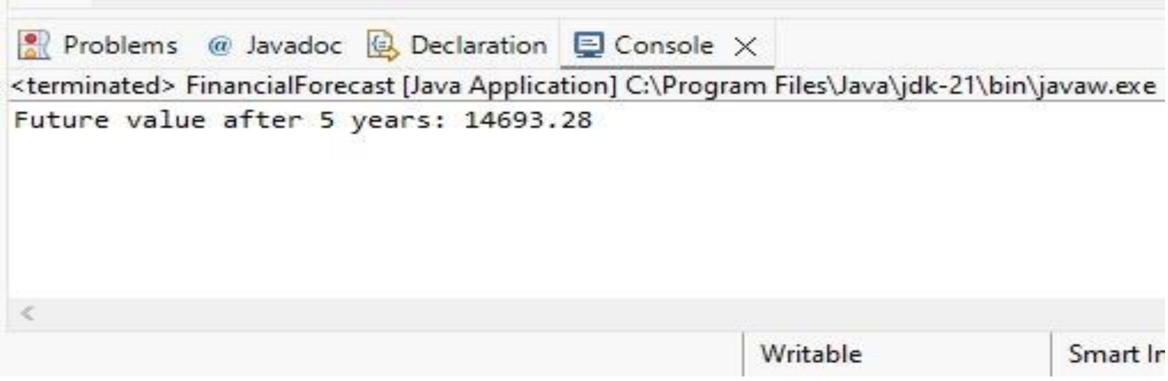
```
        if (years == 0) {
            return initialValue;
        }
        return (1 + growthRate) * futureValue(initialValue, growthRate, years - 1);
    }
    public static void main(String[] args) {
        double currentValue = 10000.0;
        double rate = 0.08;
        int years = 5;

        double result = futureValue(currentValue, rate, years);
        System.out.printf("Future value after %d years: %.2f\n", years, result);
    }
}
```

OPTIMIZEDFORECAST.JAVA

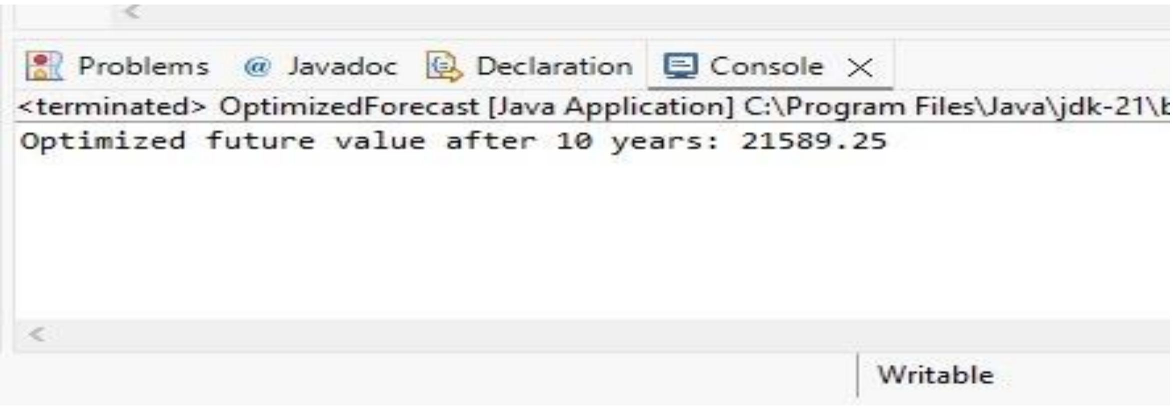
```
package FF;
import java.util.HashMap;
import java.util.Map;
public class OptimizedForecast {
               private static Map<Integer, Double> memo = new HashMap<>();
                  public static double futureValue(double initialValue, double growthRate, int years) {
                      if (years == 0) return initialValue;
                      if (memo.containsKey(years)) return memo.get(years);
                      double result = (1 + growthRate) * futureValue(initialValue, growthRate, years - 1);
                      memo.put(years, result);
                      return result;
                  }
                  public static void main(String[] args) {
                      double currentValue = 10000;
                      double rate = 0.08;
                      int years = 10;
```

```
            double result = futureValue(currentValue, rate, years);
            System.out.printf("Optimized future value after %d years: %.2f\n", years, result);
        }
}
```

OUTPUT



```
<terminated> FinancialForecast [Java Application] C:\Program Files\Java\jdk-21\bin\javaw.exe
Future value after 5 years: 14693.28
```



```
<terminated> OptimizedForecast [Java Application] C:\Program Files\Java\jdk-21\b
Optimized future value after 10 years: 21589.25
```