**Module 1: DATA STRUCTURES AND ALGORITHMS**

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

**Solution:**

**Step 1:** Understand Asymptotic notation

Big O describes the speed such as how fast or slow an algorithm

**Step 2:** Setup & Step 3: Implementation

**CODE:**

import java.util.\*;

public class Main {

static class Product {

int id;

String name;

Product(int id, String name) {

this.id = id;

this.name = name;

}

public String toString() {

return "Product ID: " + id + ", Name: " + name;

}

}

public static void main(String[] args) {

Product[] products = {

new Product(1, "Phone"),

new Product(2, "Laptop"),

new Product(3, "Shoes"),

new Product(4, "Watch"),

new Product(5, "Camera")

};

Scanner sc = new Scanner(System.in);

System.out.print("Enter product name to search: ");

String search = sc.nextLine();

boolean found = false;

for (Product p : products) {

if (p.name.equalsIgnoreCase(search)) {

System.out.println("Found (Linear Search): " + p);

found = true;

break;

}

}

if (!found) System.out.println("Not found (Linear Search)");

Arrays.sort(products, Comparator.comparing(p -> p.name.toLowerCase()));

int low = 0, high = products.length - 1;

found = false;

while (low <= high) {

int mid = (low + high) / 2;

int cmp = search.compareToIgnoreCase(products[mid].name);

if (cmp == 0) {

System.out.println("Found (Binary Search): " + products[mid]);

found = true;

break;

} else if (cmp < 0) {

high = mid - 1;

} else {

low = mid + 1;

}

}

if (!found) System.out.println("Not found (Binary Search)");

}

}

**A screenshot of a computer

AI-generated content may be incorrect.OUTPUT:**

**Step 4:**

Linear search takes O(n) time.

Binary Search uses divide-and-conquer and takes O(log n) and works only if array is sorted

For platforms like E-commerce binary search is more suitable because it has more products and searching needs to be fast.

**Exercise 7: Financial Forecasting**

**Step 1:** Understanding Recursive Algorithms

A method calls itself to solve smaller parts

**Step 2:** Setup & Step 3: Implementation

CODE:

import java.util.Scanner;

public class Main {

public static double futureValue(double amount, double rate, int years) {

if (years == 0) {

return amount;

}

return futureValue(amount, rate, years - 1) \* (1 + rate);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("current amount: ");

double amount = sc.nextDouble();

System.out.print("annual growth rate: ");

double rate = sc.nextDouble();

System.out.print("number of years: ");

int years = sc.nextInt();

double future = futureValue(amount, rate, years);

System.out.printf("Future value after %d years: %.2f\n", years, future);

}

}

**OUTPUT:**

A screen shot of a computer

AI-generated content may be incorrect.

**Step 4:** Time Complexity & Optimization

Time Complexity : O(n)

Use memoization or rewrite it as iterative to avoid call stack overhead.