ALGORITHMS AND DATA STRUCTURES – OUTPUTS

Exercise 1: Inventory Management System

```
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\inventory> java Main.java
Product added.
Product added.
Product[ID=P001, Name=Laptop, Qty=10, Price=55000.0]
Product[ID=P002, Name=Mouse, Qty=50, Price=500.0]
Product updated.
Product deleted.
Product[ID=P001, Name=Laptop, Qty=8, Price=53000.0]
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\inventory>
```

1.Why are data structures and algorithms important here?

Efficient inventory management involves fast lookup, updates, and deletions. Without the right data structure, these operations could become slow and impact performance as inventory grows

2.Which data structures are suitable?

HashMap: Maps productId → Product, giving O(1) time for add, update, delete, and search

3.Analysis

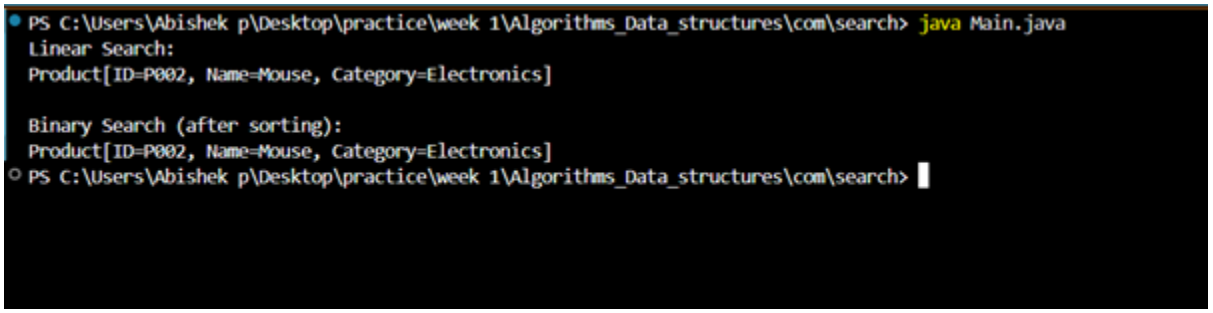
Operation	HashMap Time Complexity
Add Product	O(1) average case
Update Product	O(1) average case
Delete Product	O(1) average case
View Inventory	O(n) (where n = number of products)

4.optimizations

If the product ID is known, HashMap gives constant time operations ,to sort or search by other fields like productName,

you could use a secondary data structure or apply sorting/searching techniques as needed

Exercise 2: E-commerce Platform Search Function



1.Time Complexity:

Operation	Linear Search	Binary Search (sorted)
Best Case	O(1)	O(1)
Average Case	O(n)	O(log n)
Worst Case	O(n)	O(log n)

2.When to Use:

Use linear search if the list is unsorted or small.

Use binary search if the list is sorted and performance is critical.

Exercise 3: Sorting Customer Orders

```
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\sorting> java Main.java
Original Orders:
Order[ID=O101, Customer=Alice, Price=3000.5]
Order[ID=O102, Customer=Bob, Price=1500.75]
Order[ID=O103, Customer=Charlie, Price=4500.0]
Order[ID=O104, Customer=Daisy, Price=2000.0]

Bubble Sort by totalPrice (descending):
Order[ID=O103, Customer=Charlie, Price=4500.0]
Order[ID=O101, Customer=Alice, Price=3000.5]
Order[ID=O104, Customer=Daisy, Price=2000.0]
Order[ID=O102, Customer=Bob, Price=1500.75]

Quick Sort by totalPrice (descending):
Order[ID=O103, Customer=Charlie, Price=4500.0]
Order[ID=O101, Customer=Alice, Price=3000.5]
Order[ID=O104, Customer=Daisy, Price=2000.0]
Order[ID=O102, Customer=Bob, Price=1500.75]
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\sorting>
```

1.Time Complexity Comparison

Algorithm	Best Case	Average Case	Worst Case
Bubble Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Quick Sort	O(n log n)	O(n log n)	$O(n^2)$

Exercise 4: Employee Management System

```
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\employee> java Main.java
Employee added.
Employee added.
Employee added.

All Employees:
Employee[ID=E001, Name=Alice, Position=Developer, Salary=70000.0]
Employee[ID=E002, Name=Bob, Position=Manager, Salary=90000.0]
Employee[ID=E003, Name=Charlie, Position=Tester, Salary=50000.0]

Searching for E002:
Employee[ID=E002, Name=Bob, Position=Manager, Salary=90000.0]

Deleting E001:
?? Employee deleted.

After Deletion:
Employee[ID=E002, Name=Bob, Position=Manager, Salary=90000.0]
Employee[ID=E003, Name=Charlie, Position=Tester, Salary=50000.0]
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\employee>
```

1.Time Complexity Analysis

Operation	Time Complexity
Add	O(1)
Search	O(n)
Delete	O(n)
Traverse	O(n)

Exercise 5: Task Management System

```
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\task> java Main.java
Task added.
Task added.
Task added.

All Tasks:
Task[ID=T001, Name=Fix bugs, Status=Pending]
Task[ID=T002, Name=Write docs, Status=In Progress]
Task[ID=T003, Name=Deploy app, Status=Pending]

Searching for T002:
Task[ID=T002, Name=Write docs, Status=In Progress]

Deleting T001:
Task deleted.

After Deletion:
Task[ID=T002, Name=Write docs, Status=In Progress]
Task[ID=T003, Name=Deploy app, Status=Pending]
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\task>
```

1.Time Complexity Analysis

Operation	Time Complexity
Add	O(n)
Search	O(n)
Delete	O(n)
Traverse	O(n)

2.Advantages of Linked List over Arrays

Dynamic size

Efficient insert/delete at beginning/middle

Exercise 6: Library Management System

```
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\library> java Main.java
Linear Search for 'Algorithms':
Book[ID=B002, Title=Algorithms, Author=Charlie]

Sorting books by title for binary search...

Binary Search for 'Algorithms':
Book[ID=B002, Title=Algorithms, Author=Charlie]
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\library>
```

1.Time Complexity

Search Type	Best Case	Average Case	Worst Case
Linear Search	O(1)	O(n)	O(n)
Binary Search	O(1)	O(log n)	O(log n)

2.When to Use:

Use Linear Search for small or unsorted data.

3.Use Binary Search when:

The list is already sorted

Exercise 7: Financial Forecasting

```
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\forecastings> java Main.java
Forecast using plain recursion:
Year 5 forecast: ?16105.10

Forecast using memoized recursion:
Year 5 forecast: ?16105.10
PS C:\Users\Abishek p\Desktop\practice\week 1\Algorithms_Data_structures\com\forecastings>
```

1.Time Complexity Analysis

Approach	Time Complexity
Plain Recursion	O(n)
Memoized	O(n)