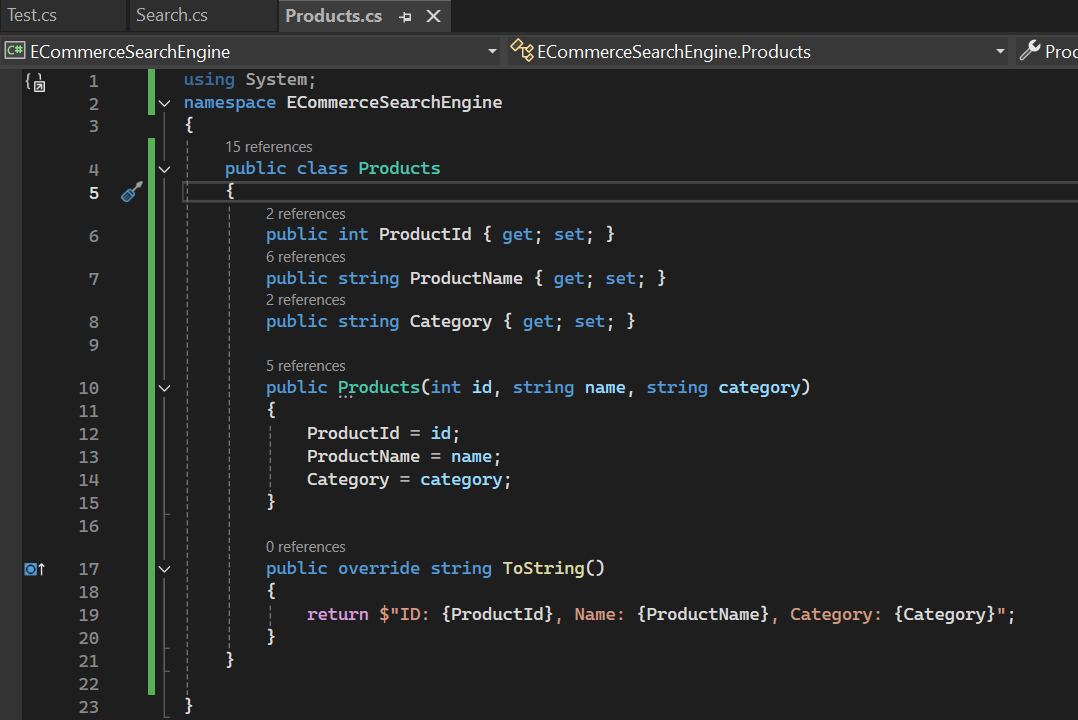
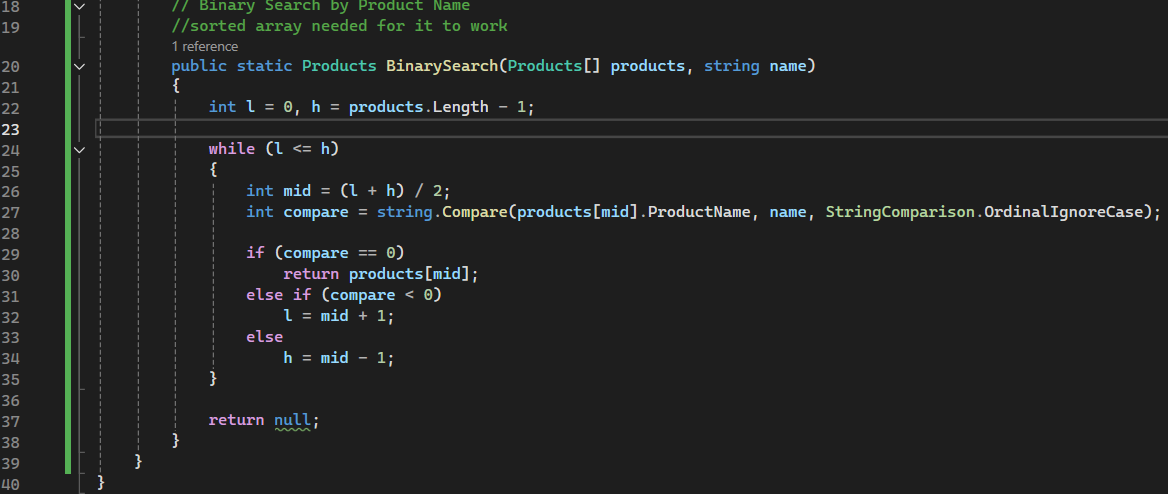
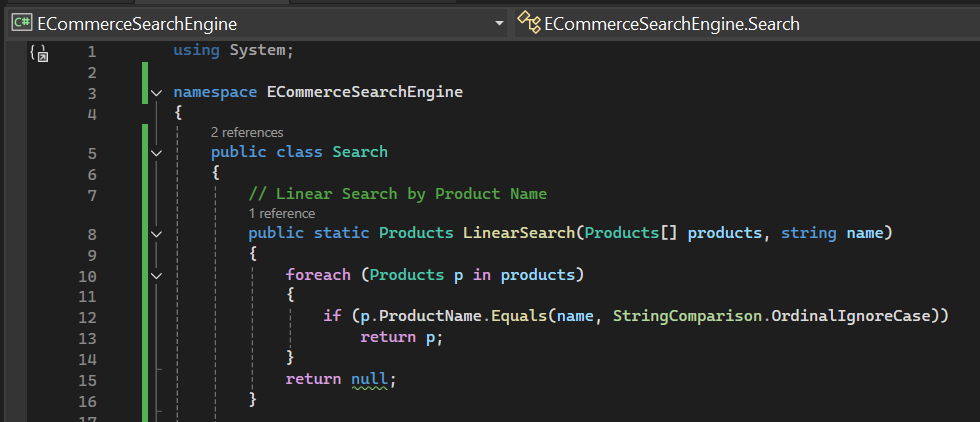
**WEEK 1 MANDATORY HANDS-ON**

**Superset ID: 6365267**

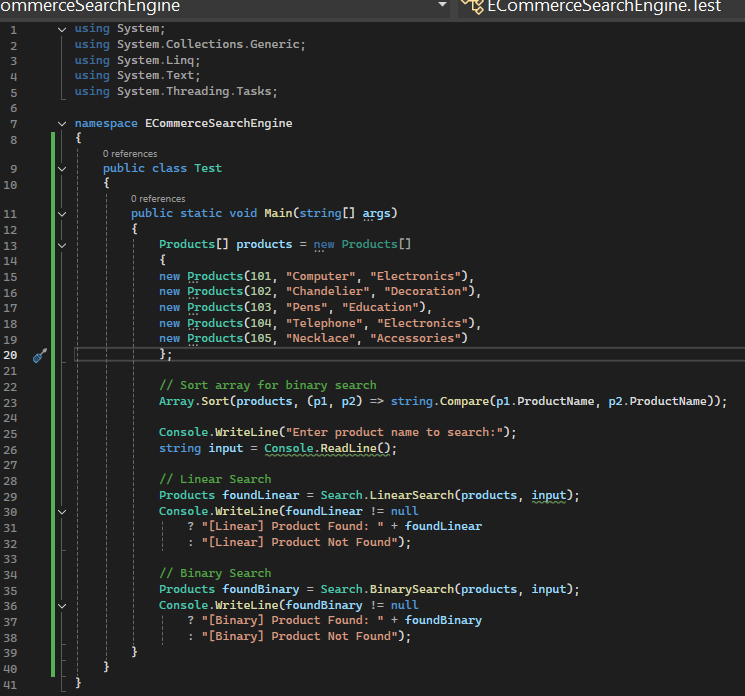
**MODULE 2: Data Structures and Algorithms**

Exercise 2: E-commerce Platform Search Function

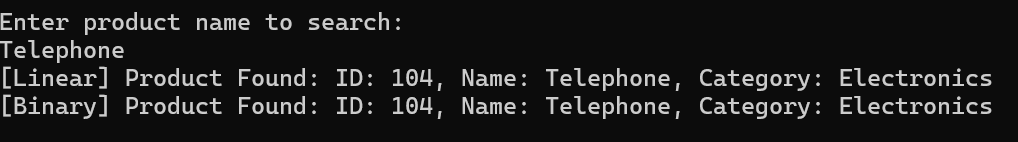
* Big O notation describes how an algorithm's runtime grows relative to input size (n). **O(1)**: Constant time (best), **O(n)**: Linear time, **O(log n):** Logarithmic time, **O(n²)**: Quadratic time etc.
* In Linear Search, the best-case time complexity is O(1) if the required item occurs at the beginning of the list that is being searched. Worst case is O(n) when the item being searched is at the end of the list or not present.
* For Binary Search, the best-case time complexity is O(1) when the required element is at the middle of the list. Worst-case time complexity is O(log n) when the list is halved until the required element is found or the search space is exhausted.
* Products class
* Linear and Binary Search Functions



Testing

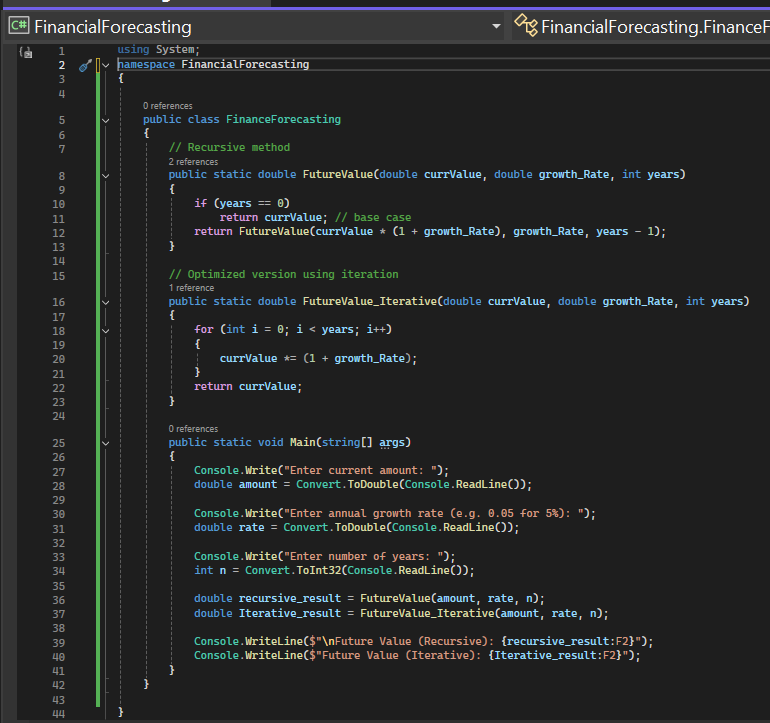


* Output



* Linear Search: No need to sort the array, simple and good for small datasets. Worst case is O(n), making it inefficient for large datasets.
* Binary Search: Sorting of the array is needed, but it is faster than linear search. Worst case is O(log n), efficient for big datasets.
* I’d recommend Linear Search if my dataset is simple and smaller. But Binary Search is ideal for large datasets, especially since we’re dealing with data from an E-Commerce website, which has to deal with a large amount of data. Binary Search offers a much faster way of searching data that is needed here.

Exercise 7: Financial Forecasting

* To solve smaller parts of a bigger problem, Recursion calls itself. It is used when a big problem can be broken into smaller sub-problems.
* **Formula:** FutureValue(years) = CurrentValue \* (1 + growthRate) ^years
* We can write this as: FV(n) = FV(n-1) \* (1 + growthRate)
* Base case: FV(0) = currentValue
* Time Complexity:
* **Recursive Version:**
  + Time complexity: **O(n)**
  + Space complexity: **O(n)**
* **Iterative Version (Optimized Method used):**
  + Time complexity: **O(n)**
  + Space complexity: **O(1)**
* Optimization: Either converting to iteration, as shown in the code, or memoization can be used as well (although not needed here).