Seatwork 6.1	
Linear and Binary Search	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 09/11/2025
Section: CPE21S4	**Date Submitted:** 09/11/2025
Name(s): Cabrera, Gabriel A.	**Instructor:** Engr. Jimlord Quejado

6. Output

Answer the following questions:

1. What is a search tree in data structures?

A tree in data structure is an abstract data type that demonstrates a hierarchy of sets of data called nodes. Each node in the tree can be connected to more than one child but each child can only be connected to one parent, except for the root node, where the set of data starts.

A search tree is a tree data structure that is utilized for locating specific data within a set. In order for a tree to be called a search tree, the key for each node must be greater than the keys in subtrees on its left side, and less keys in subtrees on the right.

2. What are the different types of search algorithm in data structures? Differentiate each type of search.

There are two main types of searching algorithms in data structures, linear search and binary search. Linear search is used for unsorted arrays, it compares the item to be searched with the array elements one by one. Its asymptotic notation is O(n) or linear time. Binary search, on the other hand, is used for an already sorted array. It compares the item to be searched with the middle element of the array first then returns it if they're the same value, otherwise, it searches either left or right of the array depending if the middle element is greater than or less than the search item. This algorithm takes O(log n) time which is faster for linear search.

3. What operations / implementations can be performed using binary and linear search operations?

Linear search is pretty simple, it starts by the user or developer picking a desired element then running a loop which iterates over a given array, comparing each element in the array with the desired element, stopping when the search element is found or when the algorithm is finished iterating over the array.
Ex.

```cpp
#include <iostream>
using namespace std;

int search(int array[], int n, int x) {

  // Going through array sequencially
  for (int i = 0; i < n; i++)
    if (array[i] == x)
      return i;
  return -1;
}

int main() {
  int array[] = {2, 4, 0, 1, 9};
  int x = 1;
  int n = sizeof(array) / sizeof(array[0]);

  int result = search(array, n, x);

  (result == -1) ? cout << "Element not found" : cout << "Element found at inde
}
```

Binary search can only be used on a sorted array, it starts by finding the middle index of the array and comparing it to the search element, if the element is found at the middle of the array, the process terminates, otherwise, it would continue searching on the left or right of the array, depending whether the search element is larger or smaller than the middle element. Its operations include searching, traversal, insertion, and deletion.

Ex.

C++ C Java Python C# JavaScript PHP

```cpp
#include <iostream>
#include <vector>
using namespace std;

int binarySearch(vector<int> &arr, int x) {
    int low = 0;
    int high = arr.size() - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;

        // Check if x is present at mid
        if (arr[mid] == x)
            return mid;

        // If x greater, ignore left half
        if (arr[mid] < x)
            low = mid + 1;

        // If x is smaller, ignore right half
        else
            high = mid - 1;
    }

    // If we reach here, then element was not present
    return -1;
}

int main() {
    vector<int> arr = { 2, 3, 4, 10, 40 };
    int x = 10;
    int result = binarySearch(arr, x);
    if(result == -1) cout << "Element is not present in
array";
    else cout << "Element is present at index " << result
    return 0;
}
```

4. What are the advantages in using binary search tree as data structure?

- Searching in binary search tree is more efficient since it has an O(log n) time complexity.
- Elements are already sorted in a binary search tree
- Insertion and deletion operations are faster in binary search trees compared to arrays and linked lists.
- Pointers and other data structures are not needed, since BST simply stores the elements.
- Since the data stored is in different nodes and arranged in a pattern, it is easier to remember the organized structure of the data.

5. Give an example program using binary search and Linear search.

Linear Search

```
#include <iostream>
using namespace std;

int search(int array[], int n, int x) {

  // Going through array sequencially
  for (int i = 0; i < n; i++)
    if (array[i] == x)
      return i;
  return -1;
}

int main() {
  int array[] = {2, 4, 0, 1, 9};
  int x = 1;
  int n = sizeof(array) / sizeof(array[0]);

  int result = search(array, n, x);

  (result == -1) ? cout << "Element not found" : cout << "Element found at inde
}
```

Binary Search

C++ C Java Python C# JavaScript PHP

```cpp
#include <iostream>
#include <vector>
using namespace std;

int binarySearch(vector<int> &arr, int x) {
    int low = 0;
    int high = arr.size() - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;

        // Check if x is present at mid
        if (arr[mid] == x)
            return mid;

        // If x greater, ignore left half
        if (arr[mid] < x)
            low = mid + 1;

        // If x is smaller, ignore right half
        else
            high = mid - 1;
    }

    // If we reach here, then element was not present
    return -1;
}

int main() {
    vector<int> arr = { 2, 3, 4, 10, 40 };
    int x = 10;
    int result = binarySearch(arr, x);
    if(result == -1) cout << "Element is not present in
array";
    else cout << "Element is present at index " << result
    return 0;
}
```

7. References

Wikipedia contributors. (n.d.). *Tree (abstract data type)*. Wikipedia. Retrieved September 11, 2025, from Tree (abstract data type) - Wikipedia

Wikipedia contributors. (n.d.). *Search tree*. Wikipedia. Retrieved September 11, 2025, from Search tree - Wikipedia

Searching Algorithms - GeeksforGeeks

GeeksforGeeks. (2025, July 23). *Searching algorithms*. GeeksforGeeks. Retrieved from Linear Search (With Code)

Programiz. (n.d.). *Linear search (with code)*. Programiz. Retrieved September 11, 2025, from Binary Search - GeeksforGeeks

TPointTech. (n.d.). *Advantages and disadvantages of binary search tree*. Retrieved September 11, 2025, from Advantages and Disadvantages of Binary Search Tree - Tpoint Tech

GeeksforGeeks. (n.d.). *Applications, advantages and disadvantages of binary search tree*. Retrieved September 11, 2025, from Applications, Advantages and Disadvantages of Binary Search Tree - GeeksforGeeks

Kumari, A. (2025, March). *Binary search tree*. Netaji Subhas University of Technology. Retrieved from B.-Tech-3rd-binary-search-tree-Anjoo-Kumari.pdf

EDUCBA. (n.d.). *Binary search tree advantages*. Retrieved September 11, 2025, from Binary Search Tree Advantages | Various Binary Search Tree Advantages

8. Conclusion

The activity has taught me about searching algorithms and how to implement them, especially binary search trees. It also taught me the difference of binary search trees and how they are more advantageous compared to other data structures.

While it has a lot of advantages, it also comes with disadvantages which needed the array to be sorted first, memory management, and need for careful implementation. Overall, it showed that even if we can see that a data structure has many benefits, it would take a good developer to determine which one would be best to be implemented in various cases.

9. Assessment Rubric