

Hands-on Activity 6.1

Searching Techniques

Course Code: CPE010

Program: Computer Engineering

Course Title: Data Structures and Algorithms

Date Performed: 09/15/25

Section: CPE 010-CPE21S4

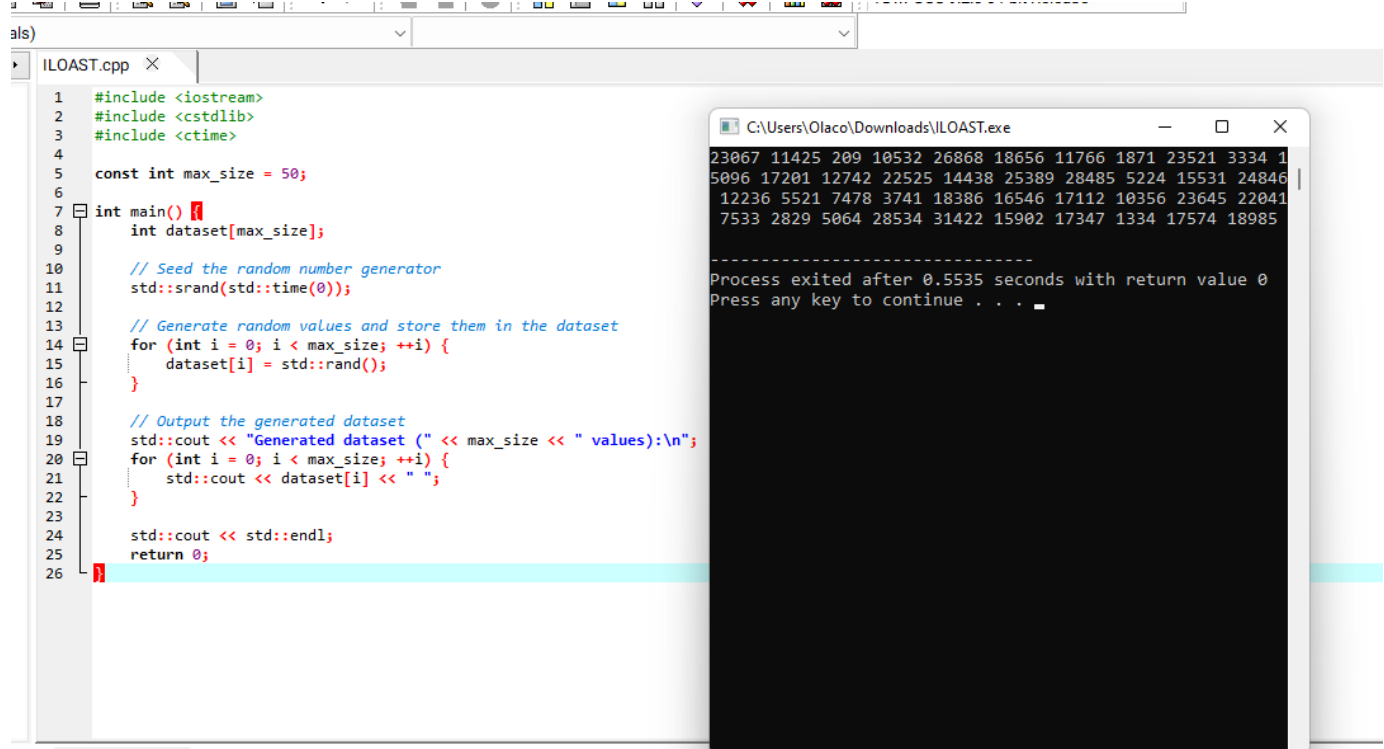
Date Submitted: 09/15/25

Name(s): CATUNGAL, KERWIN JAN B.

Instructor: Jimlord Quejado

6. Output

ILO A:



The screenshot shows a C++ program named ILOAST.cpp in a code editor. The program generates 50 random numbers and prints them. A separate window shows the output of the program, displaying the generated numbers in a grid format.

```
1 #include <iostream>
2 #include <cstdlib>
3 #include <ctime>
4
5 const int max_size = 50;
6
7 int main()
8 {
9     int dataset[max_size];
10
11     // Seed the random number generator
12     std::srand(std::time(0));
13
14     // Generate random values and store them in the dataset
15     for (int i = 0; i < max_size; ++i) {
16         dataset[i] = std::rand();
17     }
18
19     // Output the generated dataset
20     std::cout << "Generated dataset (" << max_size << " values):\n";
21     for (int i = 0; i < max_size; ++i) {
22         std::cout << dataset[i] << " ";
23     }
24
25     std::cout << std::endl;
26     return 0;
27 }
```

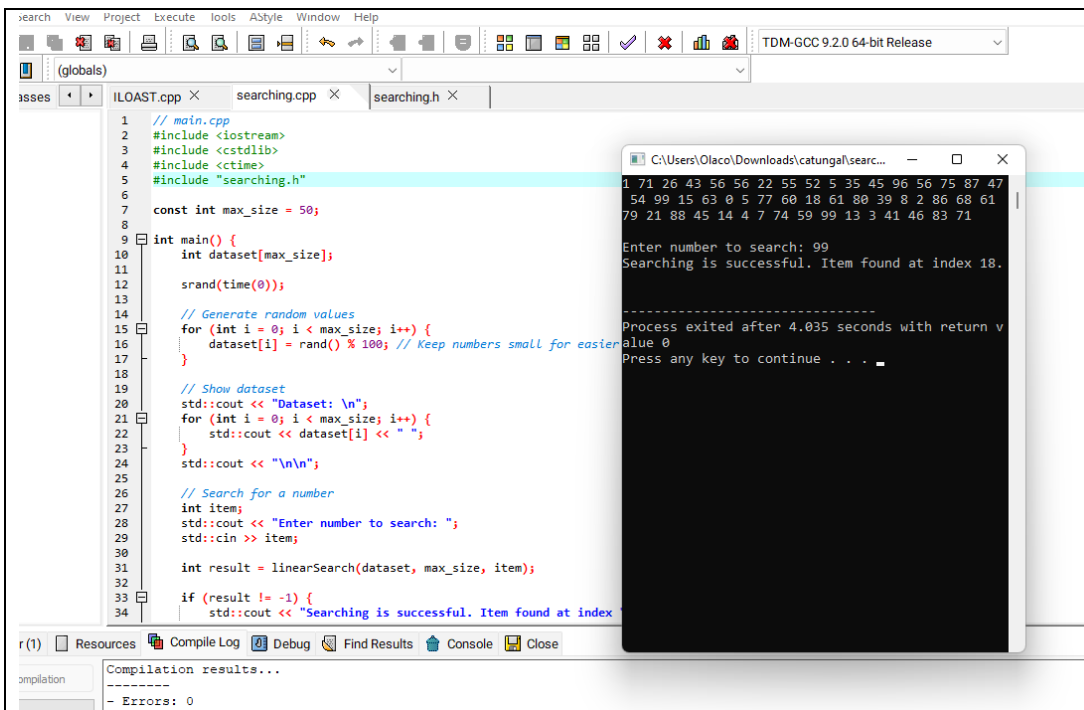
Output:

```
C:\Users\Olaco\Downloads\ILOAST.exe
23067 11425 209 10532 26868 18656 11766 1871 23521 3334 1
5096 17201 12742 22525 14438 25389 28485 5224 15531 24846
12236 5521 7478 3741 18386 16546 17112 10356 23645 22041
7533 2829 5064 28534 31422 15902 17347 1334 17574 18985

-----
Process exited after 0.5535 seconds with return value 0
Press any key to continue . . .
```

Observation: This code creates an array of 50 random numbers using rand() and prints them on the screen.

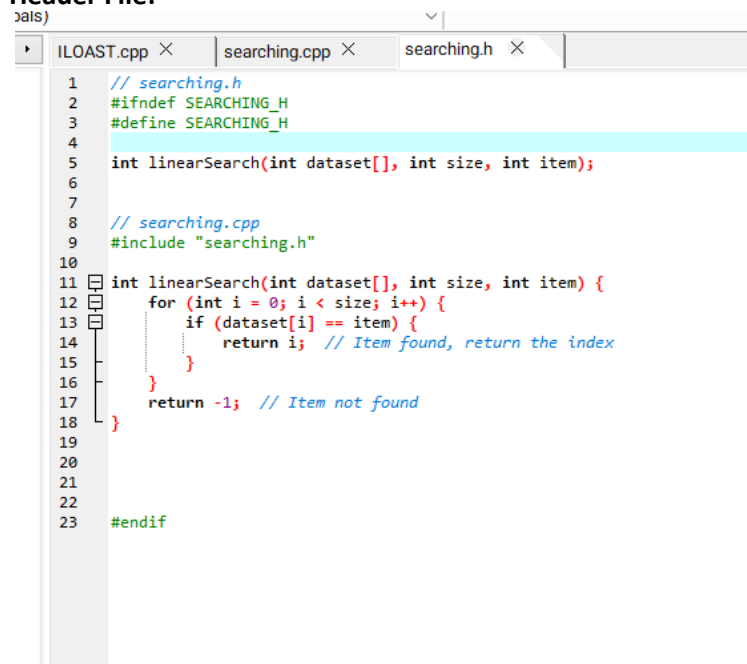
6.2 A:



```
1 // main.cpp
2 #include <iostream>
3 #include <cstdlib>
4 #include <ctime>
5 #include "searching.h"
6
7 const int max_size = 50;
8
9 int main() {
10     int dataset[max_size];
11
12     srand(time(0));
13
14     // Generate random values
15     for (int i = 0; i < max_size; i++) {
16         dataset[i] = rand() % 100; // Keep numbers small for easier search
17     }
18
19     // Show dataset
20     std::cout << "Dataset: \n";
21     for (int i = 0; i < max_size; i++) {
22         std::cout << dataset[i] << " ";
23     }
24     std::cout << "\n\n";
25
26     // Search for a number
27     int item;
28     std::cout << "Enter number to search: ";
29     std::cin >> item;
30
31     int result = linearSearch(dataset, max_size, item);
32
33     if (result != -1) {
34         std::cout << "Searching is successful. Item found at index " << result << "\n";
35     } else {
36         std::cout << "Item not found.\n";
37     }
38 }
```

```
1 71 26 43 56 56 22 55 52 5 35 45 96 56 75 87 47
2 54 99 15 63 0 5 77 60 18 61 80 39 8 2 86 68 61
3 79 21 88 45 14 4 7 74 59 99 13 3 41 46 83 71
4
5 Enter number to search: 99
6 Searching is successful. Item found at index 18.
7
8 -----
9 Process exited after 4.035 seconds with return value 0
10 Press any key to continue . . .
```

Header File:



```
1 // searching.h
2 #ifndef SEARCHING_H
3 #define SEARCHING_H
4
5 int linearSearch(int dataset[], int size, int item);
6
7
8 // searching.cpp
9 #include "searching.h"
10
11 int linearSearch(int dataset[], int size, int item) {
12     for (int i = 0; i < size; i++) {
13         if (dataset[i] == item) {
14             return i; // Item found, return the index
15         }
16     }
17     return -1; // Item not found
18 }
19
20
21
22
23 #endif
```

6.2 B:

```
ils)
ILOAST.cpp x searching.cpp x searching.h x 6-2B.cpp x linkedlist.h x
1 #include <iostream>
2 #include "linkedlist.h"
3
4 int main() {
5     Node<char>* name1 = new_node('K');
6     Node<char>* name2 = new_node('e');
7     Node<char>* name3 = new_node('r');
8     Node<char>* name4 = new_node('w');
9     Node<char>* name5 = new_node('i');
10    Node<char>* name6 = new_node('n');
11
12    name1->next = name2;
13    name2->next = name3;
14    name3->next = name4;
15    name4->next = name5;
16    name5->next = name6;
17    name6->next = nullptr;
18
19    std::cout << "Linked list (Your name): ";
20    Node<char>* temp = name1;
21    while (temp != nullptr) {
22        std::cout << temp->data << " ";
23        temp = temp->next;
24    }
25    std::cout << "\n";
26
27    char findChar;
28    std::cout << "Enter a character to search in your name: ";
29    std::cin >> findChar;
30
31    if (linearS(name1, findChar)) {
32        std::cout << "Searching is successful. " << findChar;
33    } else {
34        std::cout << "Searching is unsuccessful. " << findChar;
35    }
36}
```

C:\Users\Olaco\Downloads\6-2B.exe

Enter a character to search in your name: e
Searching is successful. 'e' is in the list.

Process exited after 2.203 seconds with return value 0
Press any key to continue . . .

Header file:

```
1 #ifndef LINKEDLIST_H
2 #define LINKEDLIST_H
3
4
5 template <typename T>
6 struct Node {
7     T data;
8     Node* next;
9 };
10
11 /
12 template <typename T>
13 Node<T>* new_node(T val) {
14     Node<T>* node = new Node<T>;
15     node->data = val;
16     node->next = nullptr;
17     return node;
18 }
19
20
21 template <typename T>
22 bool linearS(Node<T>* head, T target) {
23     Node<T>* current = head;
24     while (current != nullptr) {
25         if (current->data == target) {
26             return true;
27         }
28         current = current->next;
29     }
30     return false;
31 }
32
33 #endif S
```

Observation:

Here I created a linked list of characters, then the program lets me enter a character to check if it's found in the list.

6.3 A:

The screenshot shows a C++ IDE with two tabs: `Main_6.3A.cpp` and `searching_6.3A.h`. The `Main_6.3A.cpp` file contains the following code:

```
1 #include <iostream>
2 #include <cstdlib>
3 #include <ctime>
4 #include <algorithm>
5 #include "searching_6.3A.h"
6
7 const int max_size = 50;
8
9 int main() {
10     int dataset[max_size];
11     srand(time(0));
12
13     for (int i = 0; i < max_size; i++) {
14         dataset[i] = rand() % 100;
15     }
16
17     std::sort(dataset, dataset + max_size);
18
19     std::cout << "Sorted dataset: \n";
20     for (int i = 0; i < max_size; i++) {
21         std::cout << dataset[i] << " ";
22         if ((i + 1) % 10 == 0) std::cout << "\n";
23     }
24     std::cout << "\n";
25
26     int item;
27     std::cout << "Enter number to search: ";
28     std::cin >> item;
29
30     int result = binarySearch(dataset, max_size, item);
31
32     if (result != -1) {
33         std::cout << "Searching is successful. Item found at index " << result << ".\n";
34     } else {
35         // Item not found
36     }
37 }
```

The output window shows the sorted dataset and the search result:

```
Sorted dataset:
0 0 1 1 4 7 11 11 15 16
17 17 19 22 27 28 31 32 35 36
39 46 46 49 50 51 54 55 56 58
59 60 64 65 65 65 66 70 71 74
80 81 82 83 84 86 90 94 96 96

Enter number to search:
```

The compilation results show 0 errors and 0 warnings.

Header file:

The screenshot shows the header file `searching_6.3A.h` with the following code:

```
1 #ifndef SEARCHING_H
2 #define SEARCHING_H
3
4 int binarySearch(int arr[], int size, int item) {
5     int low = 0;
6     int high = size - 1;
7
8     while (low <= high) {
9         int mid = low + (high - low) / 2;
10
11         if (arr[mid] == item) {
12             return mid;
13         }
14
15         if (arr[mid] < item) {
16             low = mid + 1;
17         } else {
18             high = mid - 1;
19         }
20     }
21
22     return -1; // Item not found
23 }
24
25 #endif
```

Observation: here it generates random numbers, sorts them, and then uses binary search to check if the number I enter is in the list.

6.3 B:

```
Main_6.3B.cpp × searching_6.3B.h × nodes.h ×
1 #include <iostream>
2 #include "nodes.h"
3 #include "searching_6.3B.h"
4
5 int main() {
6     Node<int>* head = new_node(10);
7     Node<int>* node2 = new_node(20);
8     Node<int>* node3 = new_node(30);
9     Node<int>* node4 = new_node(40);
10    Node<int>* node5 = new_node(50);
11
12    head->next = node2;
13    node2->next = node3;
14    node3->next = node4;
15    node4->next = node5;
16
17    std::cout << "Linked list content: ";
18    Node<int>* temp = head;
19    while (temp != NULL) {
20        std::cout << temp->data << " ";
21        temp = temp->next;
22    }
23    std::cout << "\n";
24
25    int key;
26    std::cout << "Enter number to search: ";
27    std::cin >> key;
28
29    Node<int>* result = binarySearchLinkedList(head, key);
30
31    if (result != NULL) {
32        std::cout << "Searching is successful. Item " << key << " found in linked list.\n";
33    } else {
34        std::cout << "Searching is unsuccessful. Item " << key << " not found in linked list.\n";
35    }
36}
```

C:\Users\Olaco\Downloads\Main_6.3B.exe

Linked list content: 10 20 30 40 50
Enter number to search: 30
Searching is successful. Item 30 found in linked list.

Process exited after 14.98 seconds with return value 0
Press any key to continue . . .

sources Compile Log Debug Find Results Console Close

```
Main_6.3B.cpp × searching_6.3B.h × nodes.h ×
1 #ifndef SEARCHING_H
2 #define SEARCHING_H
3
4 template <typename T>
5 Node<T>* binarySearchLinkedList(Node<T>* head, T key) {
6     Node<T>* low = head;
7     Node<T>* high = nullptr;
8     Node<T>* mid = nullptr;
9
10    while (low != nullptr) {
11        mid = low;
12        high = low->next;
13
14        // Check if mid is the correct match
15        if (mid != nullptr && mid->data == key) {
16            return mid;
17        }
18
19        if (mid->data < key) {
20            low = mid->next;
21        } else {
22            high = mid;
23        }
24    }
25
26    return nullptr; // Item not found
27 }
28
29 #endif
```

```

1  #ifndef NODES_H
2  #define NODES_H
3
4  template <typename T>
5  struct Node {
6      T data;
7      Node* next;
8
9      Node(T data) : data(data), next(nullptr) {}
10 };
11
12 template <typename T>
13 Node<T>* new_node(T data) {
14     return new Node<T>(data);
15 }
16
17 #endif
18

```

7. Supplementary Activity

PROBLEM 1:

```

(globals)
Supplementary_1.cpp x sup problem 2.cpp x sup problem3.cpp x sup problem 4.cpp x
38     }
39     return -1;
40 }
41
42 int main() {
43     int arr[] = {15, 18, 2, 19, 18, 0, 8, 14, 19, 14};
44     int size = sizeof(arr) / sizeof(arr[0]);
45     int key = 18;
46     int comparisonsArray, comparisonsList;
47
48     Node* head = createList(arr, size);
49
50     int posArray = searchArray(arr, size, key, comparisonsArray);
51     int posList = searchList(head, key, comparisonsList);
52
53     if (posArray != -1)
54         cout << "Array: Found " << key << " at index " << posArray << " after " << comparisonsArray << " comparisons.\n";
55     else
56         cout << "Array: " << key << " not found after " << comparisonsArray << " comparisons.\n";
57
58     if (posList != -1)
59         cout << "Linked List: Found " << key << " at node " << posList << " after " << comparisonsList << " comparisons.\n";
60     else
61         cout << "Linked List: " << key << " not found after " << comparisonsList << " comparisons.\n";
62
63     // Clean up Linked List
64     while (head) {
65         Node* temp = head;
66         head = head->next;
67         delete temp;
68     }
69
70     return 0;
71 }

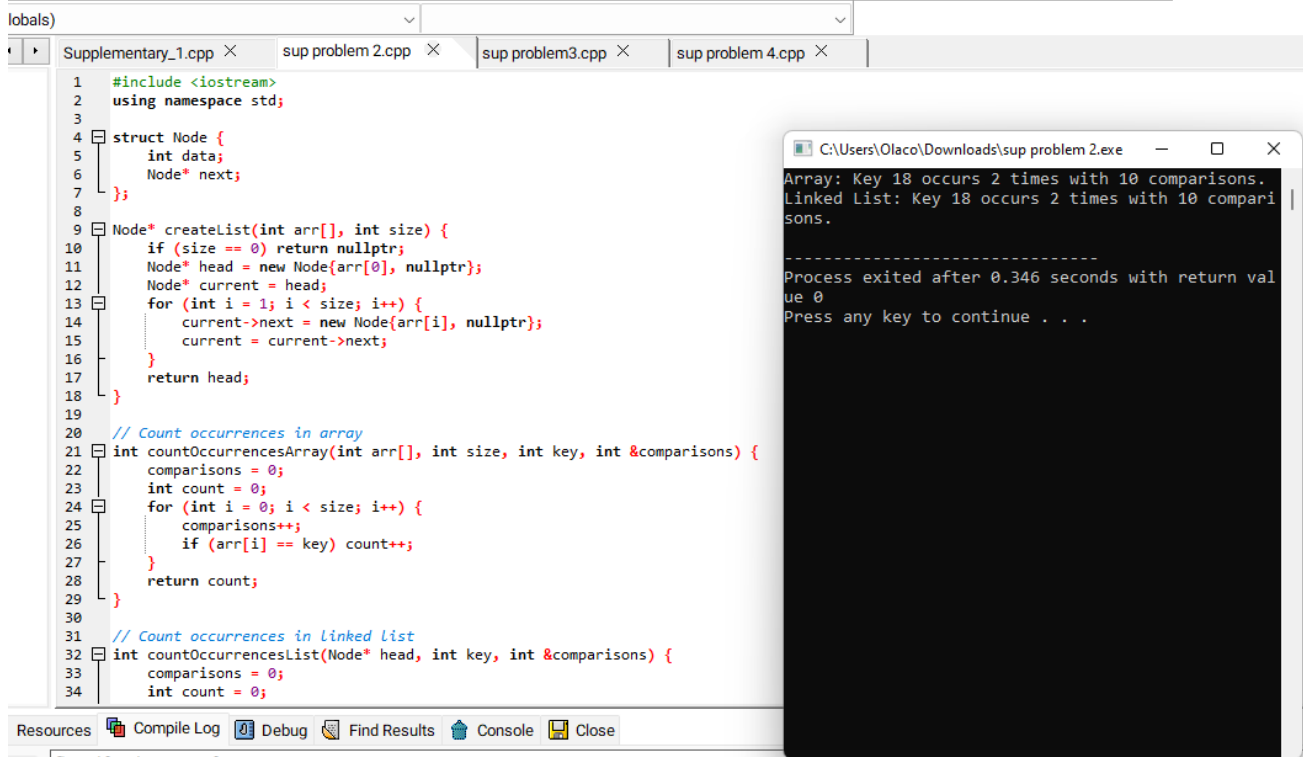
```

C:\Users\Olaco\Downloads\Supplementary_1...
 Array: Found 18 at index 1 after 2 comparisons.
 Linked List: Found 18 at node 1 after 2 comparisons.

 Process exited after 0.3631 seconds with return value 0
 Press any key to continue . . .

Here it will look for a number both in an array and in a linked list, then it shows me the position where it was found and how many steps it took to find it.

PROBLEM 2:



The screenshot shows a C++ IDE with four tabs: Supplementary_1.cpp, sup problem 2.cpp, sup problem3.cpp, and sup problem 4.cpp. The active tab is sup problem 2.cpp, which contains the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 struct Node {
5     int data;
6     Node* next;
7 };
8
9 Node* createList(int arr[], int size) {
10     if (size == 0) return nullptr;
11     Node* head = new Node(arr[0], nullptr);
12     Node* current = head;
13     for (int i = 1; i < size; i++) {
14         current->next = new Node(arr[i], nullptr);
15         current = current->next;
16     }
17     return head;
18 }
19
20 // Count occurrences in array
21 int countOccurrencesArray(int arr[], int size, int key, int &comparisons) {
22     comparisons = 0;
23     int count = 0;
24     for (int i = 0; i < size; i++) {
25         comparisons++;
26         if (arr[i] == key) count++;
27     }
28     return count;
29 }
30
31 // Count occurrences in Linked List
32 int countOccurrencesList(Node* head, int key, int &comparisons) {
33     comparisons = 0;
34     int count = 0;
```

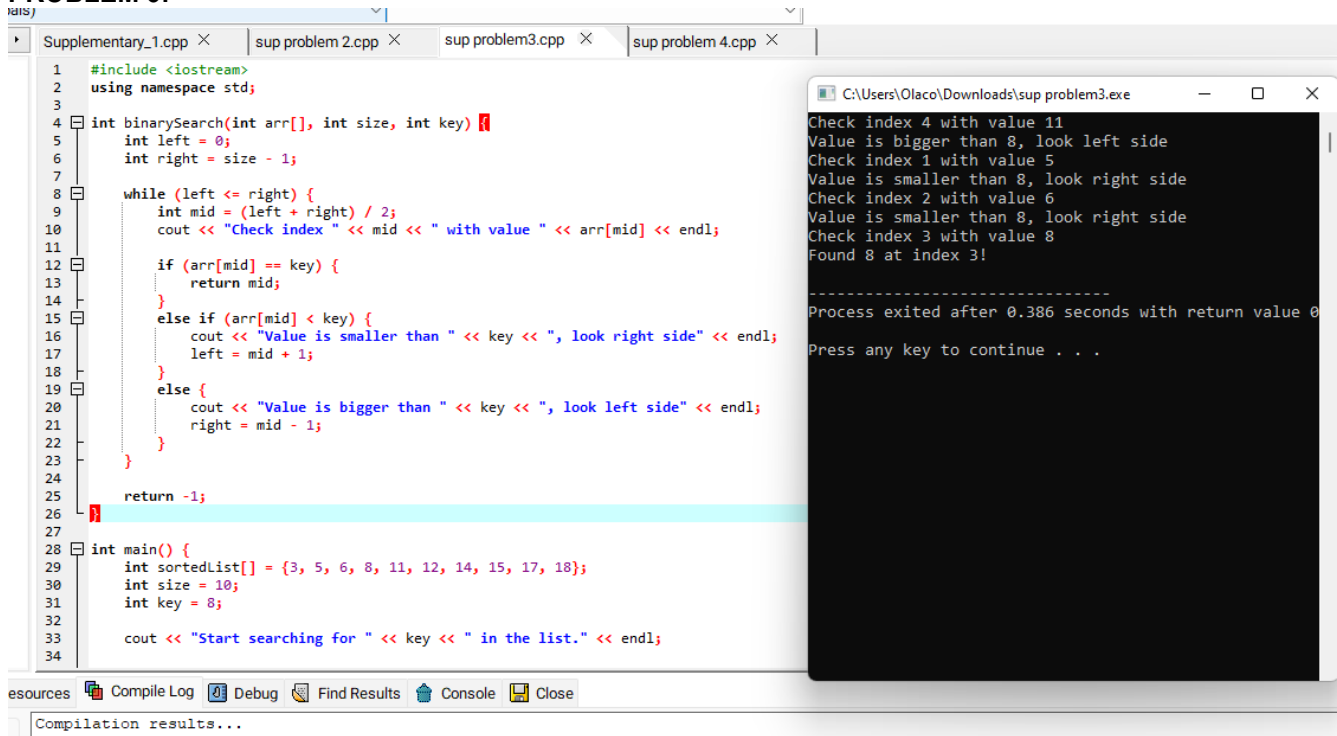
The console window shows the output of the program:

```
C:\Users\Olaco\Downloads\sup problem 2.exe
Array: Key 18 occurs 2 times with 10 comparisons.
Linked List: Key 18 occurs 2 times with 10 comparisons.

-----
Process exited after 0.346 seconds with return value 0
Press any key to continue . . .
```

It will just counts how many times a number appears in both an array and a linked list, and it also shows how many comparisons it needed to do that

PROBLEM 3:



The screenshot shows a C++ IDE with four tabs: Supplementary_1.cpp, sup problem 2.cpp, sup problem3.cpp, and sup problem 4.cpp. The active tab is sup problem3.cpp, which contains the following code:

```
1 #include <iostream>
2 using namespace std;
3
4 int binarySearch(int arr[], int size, int key) {
5     int left = 0;
6     int right = size - 1;
7
8     while (left <= right) {
9         int mid = (left + right) / 2;
10        cout << "Check index " << mid << " with value " << arr[mid] << endl;
11
12        if (arr[mid] == key) {
13            return mid;
14        }
15        else if (arr[mid] < key) {
16            cout << "Value is smaller than " << key << ", look right side" << endl;
17            left = mid + 1;
18        }
19        else {
20            cout << "Value is bigger than " << key << ", look left side" << endl;
21            right = mid - 1;
22        }
23    }
24
25    return -1;
26 }
27
28 int main() {
29     int sortedList[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
30     int size = 10;
31     int key = 8;
32
33     cout << "Start searching for " << key << " in the list." << endl;
34 }
```

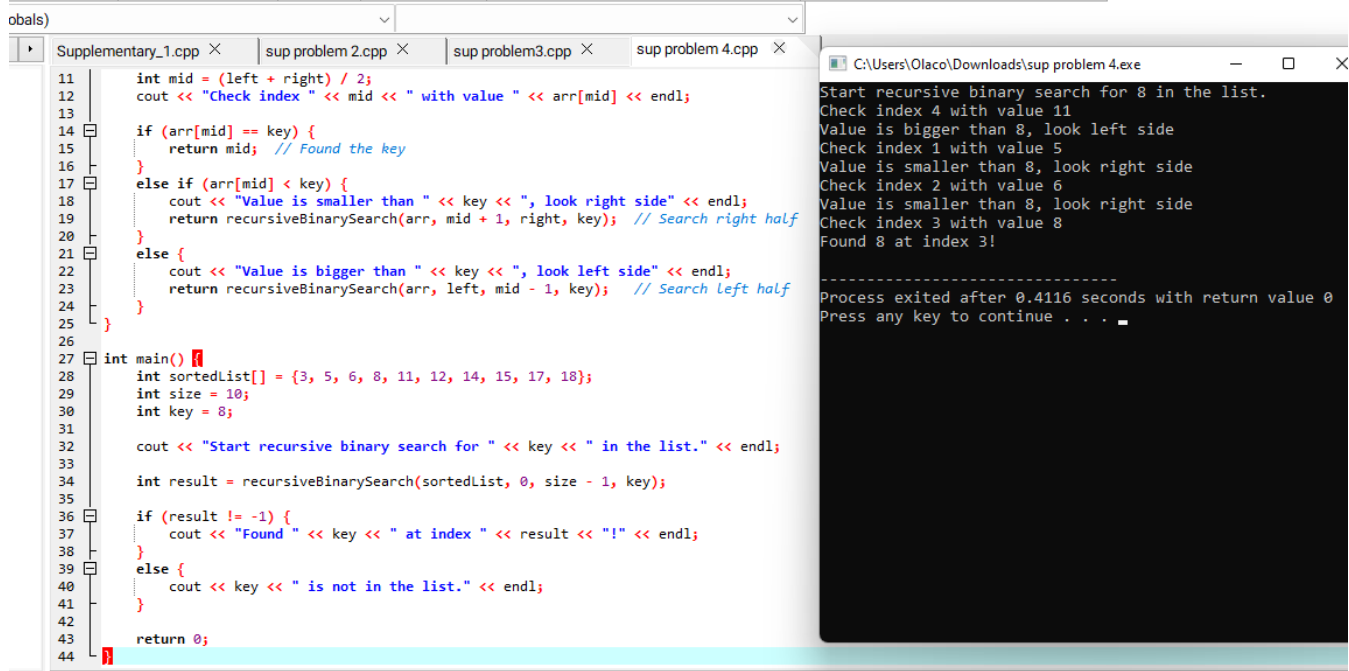
The console window shows the output of the program:

```
C:\Users\Olaco\Downloads\sup problem3.exe
Check index 4 with value 11
Value is bigger than 8, look left side
Check index 1 with value 5
Value is smaller than 8, look right side
Check index 2 with value 6
Value is smaller than 8, look right side
Check index 3 with value 8
Found 8 at index 3!

-----
Process exited after 0.386 seconds with return value 0
Press any key to continue . . .
```

Here I used binary search on a sorted list to find a number, and the program shows each step of checking until it finds the right index or says the number isn't there

PROBLEM 4:



The screenshot shows a C++ IDE with four tabs: Supplementary_1.cpp, sup problem 2.cpp, sup problem3.cpp, and sup problem 4.cpp. The active tab is sup problem 4.cpp, which contains the following code:

```
11 int mid = (left + right) / 2;
12 cout << "Check index " << mid << " with value " << arr[mid] << endl;
13
14 if (arr[mid] == key) {
15     return mid; // Found the key
16 }
17 else if (arr[mid] < key) {
18     cout << "Value is smaller than " << key << ", look right side" << endl;
19     return recursiveBinarySearch(arr, mid + 1, right, key); // Search right half
20 }
21 else {
22     cout << "Value is bigger than " << key << ", look left side" << endl;
23     return recursiveBinarySearch(arr, left, mid - 1, key); // Search left half
24 }
25 }
26
27 int main() {
28     int sortedList[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
29     int size = 10;
30     int key = 8;
31
32     cout << "Start recursive binary search for " << key << " in the list." << endl;
33
34     int result = recursiveBinarySearch(sortedList, 0, size - 1, key);
35
36     if (result != -1) {
37         cout << "Found " << key << " at index " << result << "!" << endl;
38     }
39     else {
40         cout << key << " is not in the list." << endl;
41     }
42
43     return 0;
44 }
```

The output window shows the execution of the program:

```
C:\Users\Olaco\Downloads\sup problem 4.exe
Start recursive binary search for 8 in the list.
Check index 4 with value 11
Value is bigger than 8, look left side
Check index 1 with value 5
Value is smaller than 8, look right side
Check index 2 with value 6
Value is smaller than 8, look right side
Check index 3 with value 8
Found 8 at index 3!
-----
Process exited after 0.4116 seconds with return value 0
Press any key to continue . . .
```

8. Conclusion

I learned the functioning of arrays, linked lists, and different methods of searching. In this context, I learned that in linear search the values are processed serially which is different in the case of binary search whereby the list is divided in order to locate the target faster, and in the case of linked lists data is kept in a chain. Overall, these activities really exposed me to the practical use of data structures and searching algorithms, but I also came to the understanding that I need to work on the way I structure my code and do not make so many errors in programming.

9. Assessment Rubric