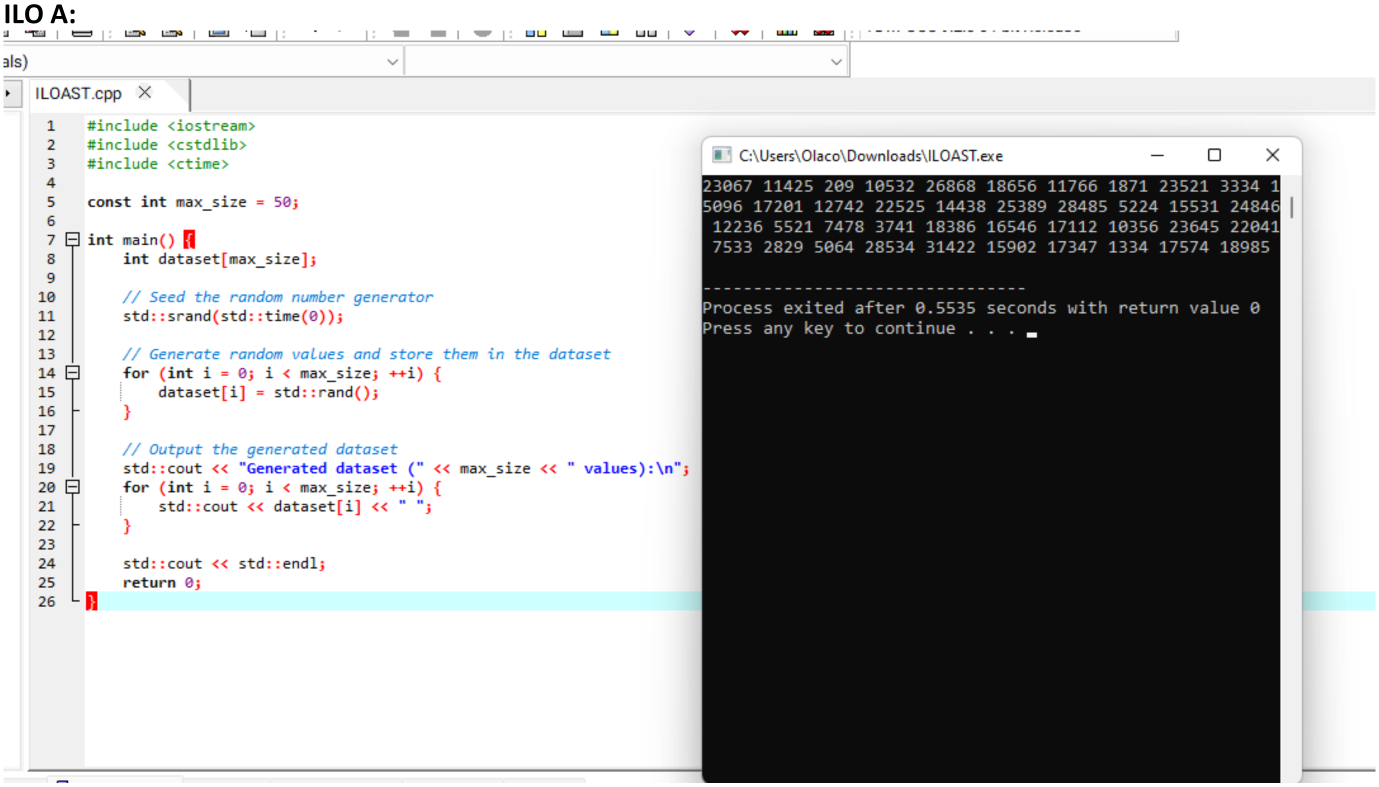
Hands-on Activity 6.1

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:

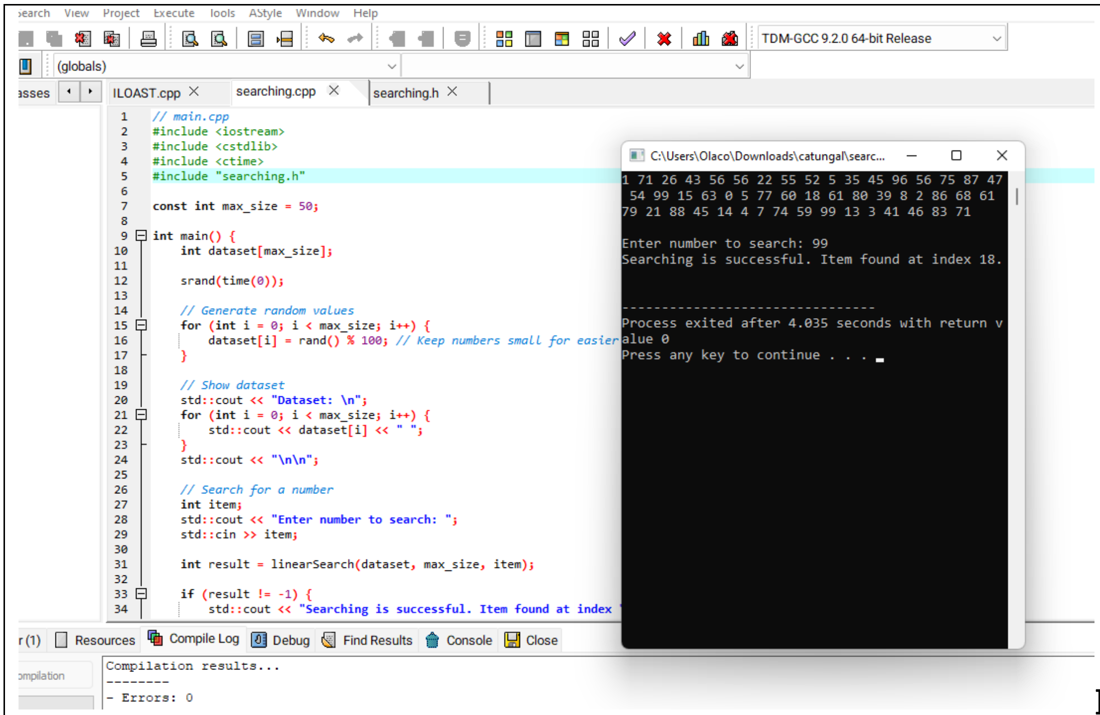


```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>

const int max_size = 50;

int main() {
    int dataset[max_size];

    // Seed the random number generator
    std::srand(std::time(0));

    // Generate random values and store them in the dataset
    for (int i = 0; i < max_size; ++i) {
        dataset[i] = std::rand();
    }

    // Output the generated dataset
    std::cout << "Generated dataset (" << max_size << " values):\n";
    for (int i = 0; i < max_size; ++i) {
        std::cout << dataset[i] << " ";
    }

    std::cout << std::endl;
    return 0;
}
```

```
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:



earch View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Release
(globals)
asses
ILOAST.cpp
searching.cpp
searching.h
C:\Users\Olaco\Downloads\catungal\searc...
r (1)
Resources
Compile Log
Debug
Find Results
Console
Close
ompilation

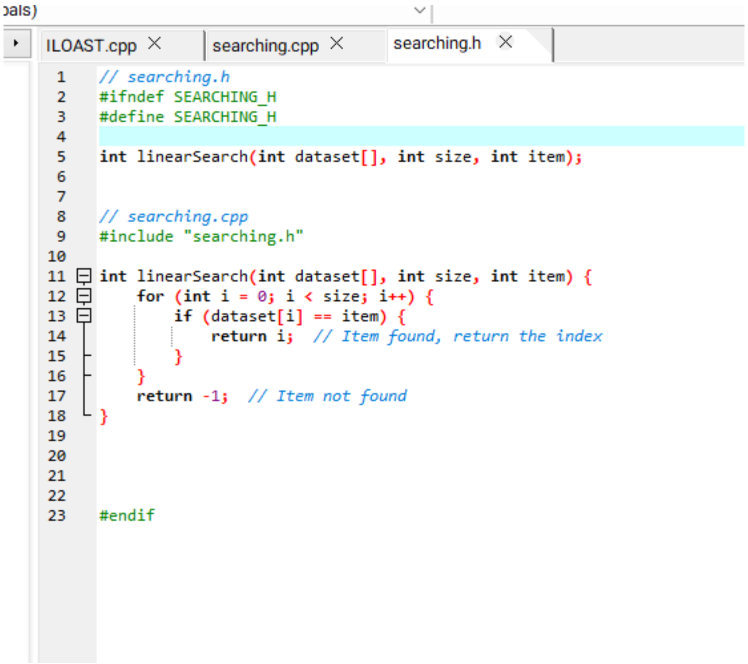
```cpp
// main.cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "searching.h"

const int max_size = 50;

int main() {
    int dataset[max_size];

    srand(time(0));

    // Generate random values
    for (int i = 0; i < max_size; i++) {
        dataset[i] = rand() % 100; // Keep numbers small for easier
    }

    // Show dataset
    std::cout << "Dataset: \n";
    for (int i = 0; i < max_size; i++) {
        std::cout << dataset[i] << " ";
    }
    std::cout << "\n\n";

    // Search for a number
    int item;
    std::cout << "Enter number to search: ";
    std::cin >> item;

    int result = linearSearch(dataset, max_size, item);

    if (result != -1) {
        std::cout << "Searching is successful. Item found at index
```

```
1 71 26 43 56 56 22 55 52 5 35 45 96 56 75 87 47
 54 99 15 63 0 5 77 60 18 61 80 39 8 2 86 68 61
79 21 88 45 14 4 7 74 59 99 13 3 41 46 83 71

Enter number to search: 99
Searching is successful. Item found at index 18.


--------------------------------
Process exited after 4.035 seconds with return v
alue 0
Press any key to continue . . .
```

```
Compilation results...
--------
- Errors: 0
```

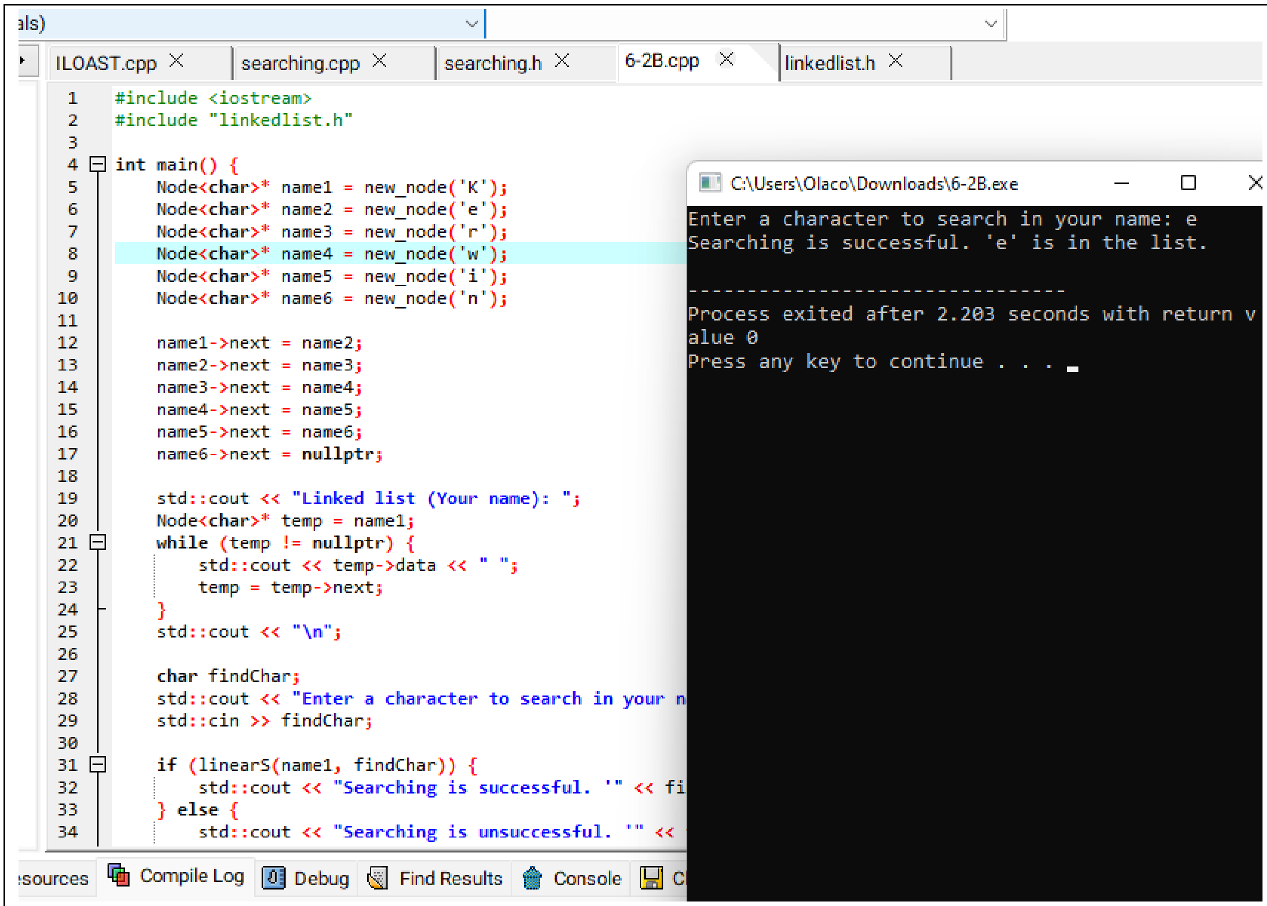
]

Header File:



```cpp
// searching.h
#ifndef SEARCHING_H
#define SEARCHING_H

int linearSearch(int dataset[], int size, int item);


// searching.cpp
#include "searching.h"

int linearSearch(int dataset[], int size, int item) {
    for (int i = 0; i < size; i++) {
        if (dataset[i] == item) {
            return i;  // Item found, return the index
        }
    }
    return -1;  // Item not found
}




#endif
```

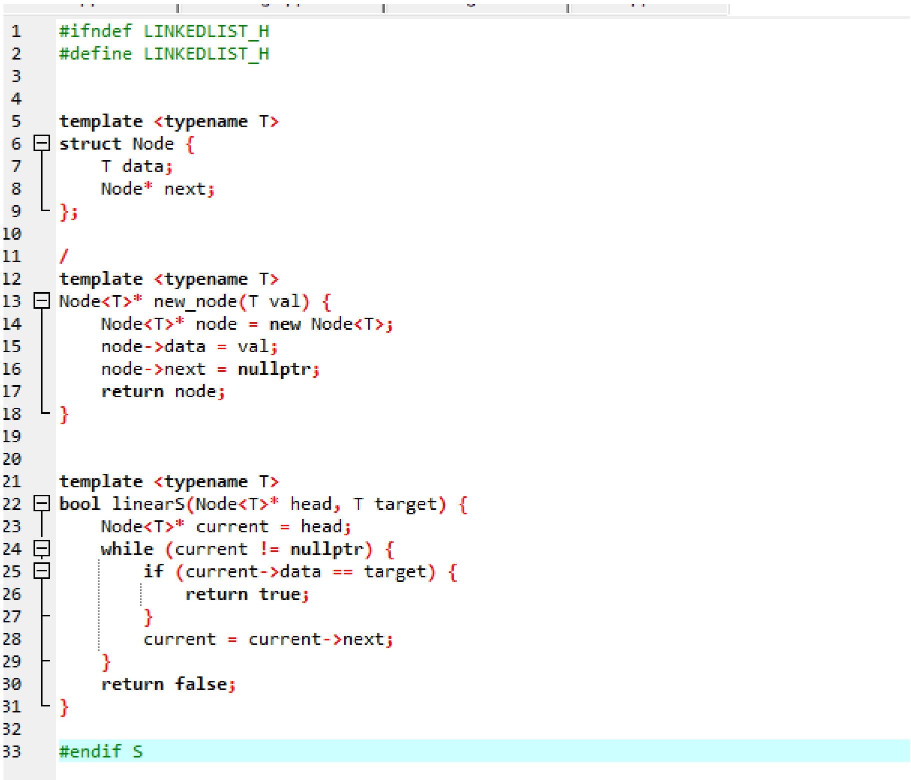
6.2 B:



```cpp
#include <iostream>
#include "linkedlist.h"

int main() {
    Node<char>* name1 = new_node('K');
    Node<char>* name2 = new_node('e');
    Node<char>* name3 = new_node('r');
    Node<char>* name4 = new_node('w');
    Node<char>* name5 = new_node('i');
    Node<char>* name6 = new_node('n');

    name1->next = name2;
    name2->next = name3;
    name3->next = name4;
    name4->next = name5;
    name5->next = name6;
    name6->next = nullptr;

    std::cout << "Linked list (Your name): ";
    Node<char>* temp = name1;
    while (temp != nullptr) {
        std::cout << temp->data << " ";
        temp = temp->next;
    }
    std::cout << "\n";

    char findChar;
    std::cout << "Enter a character to search in your n
    std::cin >> findChar;

    if (linearS(name1, findChar)) {
        std::cout << "Searching is successful. '" << fi
    } else {
        std::cout << "Searching is unsuccessful. '" << 
```

```
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 v
alue 0
Press any key to continue . . .
```

Header file:

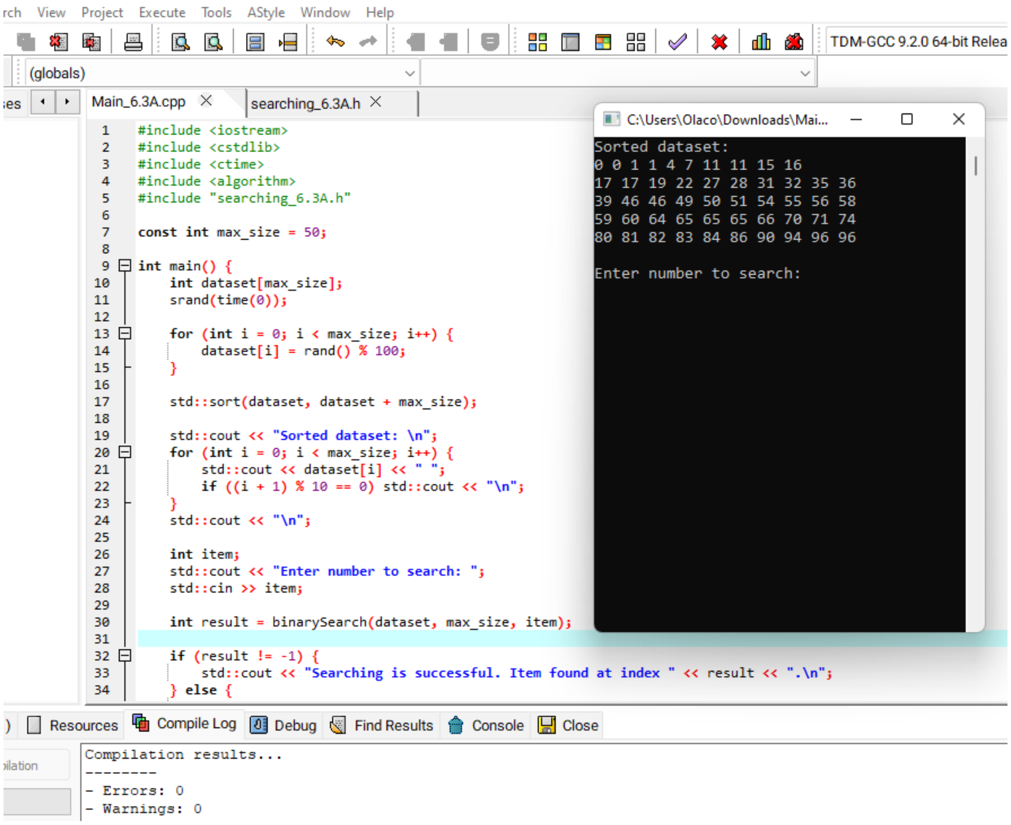


```cpp
#ifndef LINKEDLIST_H
#define LINKEDLIST_H


template <typename T>
struct Node {
    T data;
    Node* next;
};

/
template <typename T>
Node<T>* new_node(T val) {
    Node<T>* node = new Node<T>;
    node->data = val;
    node->next = nullptr;
    return node;
}


template <typename T>
bool linearS(Node<T>* head, T target) {
    Node<T>* current = head;
    while (current != nullptr) {
        if (current->data == target) {
            return true;
        }
        current = current->next;
    }
    return false;
}

#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:

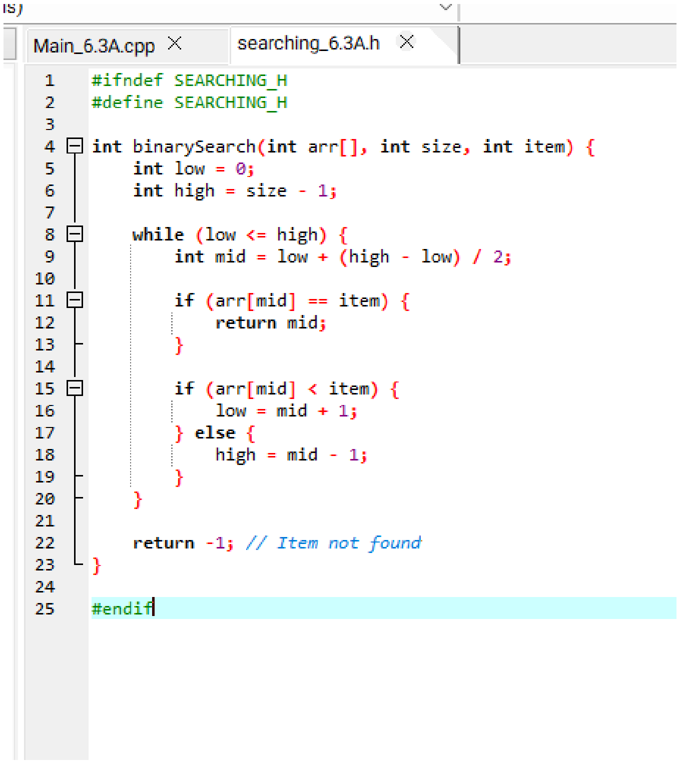

rch View Project Execute Tools AStyle Window Help
TDM-GCC 9.2.0 64-bit Relea
(globals)
Main_6.3A.cpp
searching_6.3A.h
C:\Users\Olaco\Downloads\Mai...
Resources
Compile Log
Debug
Find Results
Console
Close

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <algorithm>
#include "searching_6.3A.h"

const int max_size = 50;

int main() {
    int dataset[max_size];
    srand(time(0));

    for (int i = 0; i < max_size; i++) {
        dataset[i] = rand() % 100;
    }

    std::sort(dataset, dataset + max_size);

    std::cout << "Sorted dataset: \n";
    for (int i = 0; i < max_size; i++) {
        std::cout << dataset[i] << " ";
        if ((i + 1) % 10 == 0) std::cout << "\n";
    }
    std::cout << "\n";

    int item;
    std::cout << "Enter number to search: ";
    std::cin >> item;

    int result = binarySearch(dataset, max_size, item);

    if (result != -1) {
        std::cout << "Searching is successful. Item found at index " << result << ".\n";
    } else {
```

```
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:
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
```

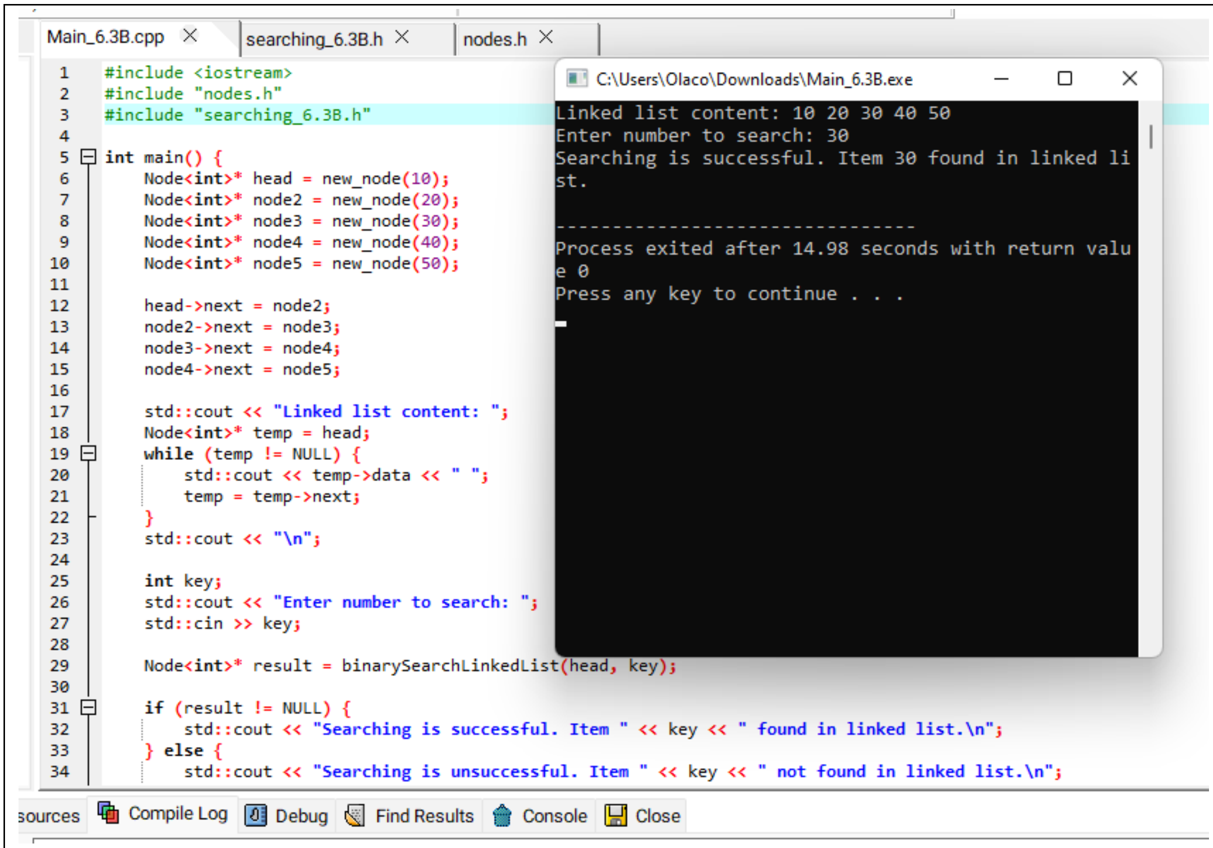
Header file:



```cpp
#ifndef SEARCHING_H
#define SEARCHING_H

int binarySearch(int arr[], int size, int item) {
    int low = 0;
    int high = size - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == item) {
            return mid;
        }

        if (arr[mid] < item) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return -1; // Item not found
}

#endif
```

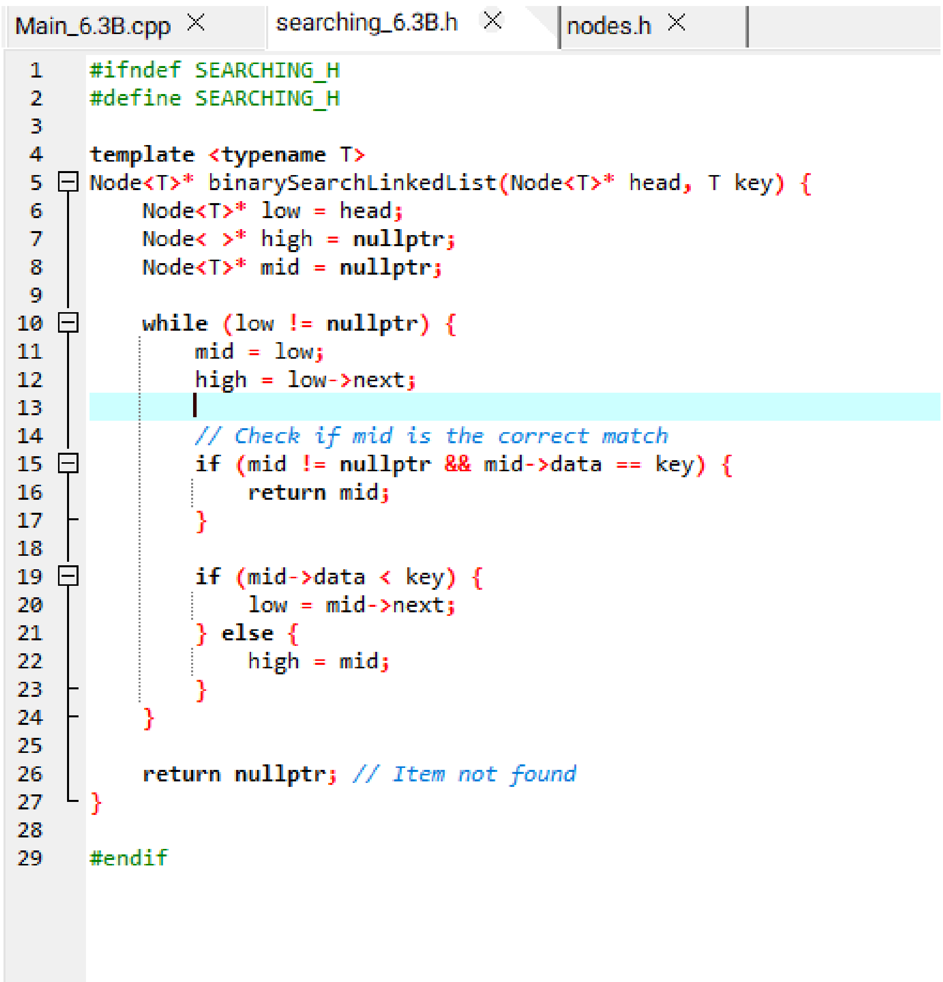
Observation: here it generates random numbers, sorts them, and then uses binary search to check if the number I enter is in the list.

6.3 B:

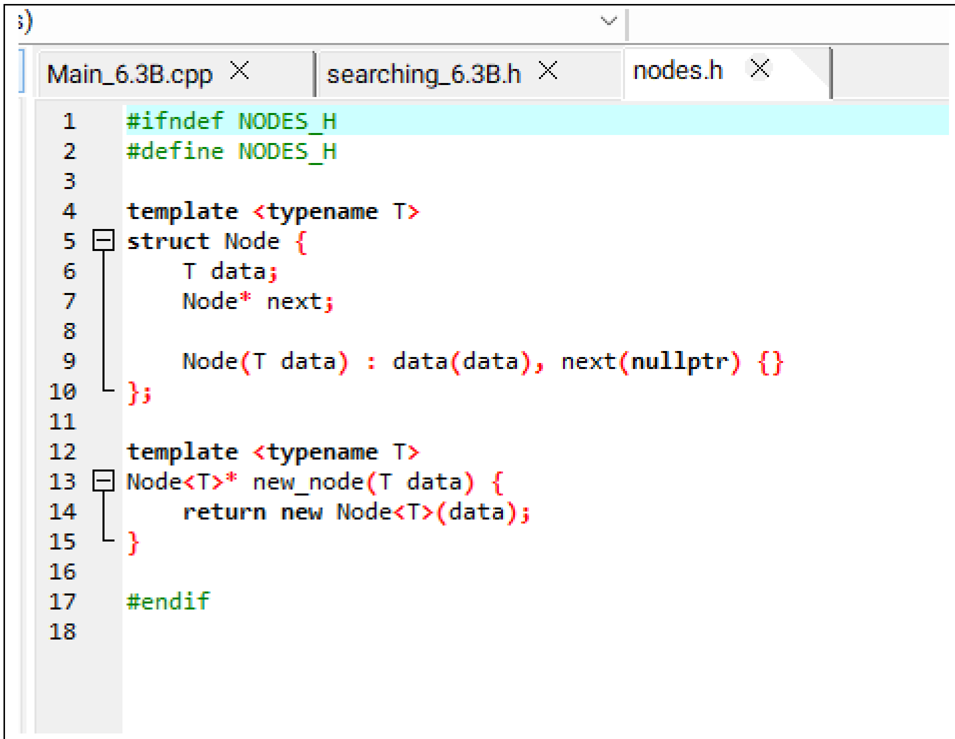


```cpp
#include <iostream>
#include "nodes.h"
#include "searching_6.3B.h"

int main() {
    Node<int>* head = new_node(10);
    Node<int>* node2 = new_node(20);
    Node<int>* node3 = new_node(30);
    Node<int>* node4 = new_node(40);
    Node<int>* node5 = new_node(50);

    head->next = node2;
    node2->next = node3;
    node3->next = node4;
    node4->next = node5;

    std::cout << "Linked list content: ";
    Node<int>* temp = head;
    while (temp != NULL) {
        std::cout << temp->data << " ";
        temp = temp->next;
    }
    std::cout << "\n";

    int key;
    std::cout << "Enter number to search: ";
    std::cin >> key;

    Node<int>* result = binarySearchLinkedList(head, key);

    if (result != NULL) {
        std::cout << "Searching is successful. Item " << key << " found in linked list.\n";
    } else {
        std::cout << "Searching is unsuccessful. Item " << key << " not found in linked list.\n";
```

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

--------------------------------
Process exited after 14.98 seconds with return valu
e 0
Press any key to continue . . .
```



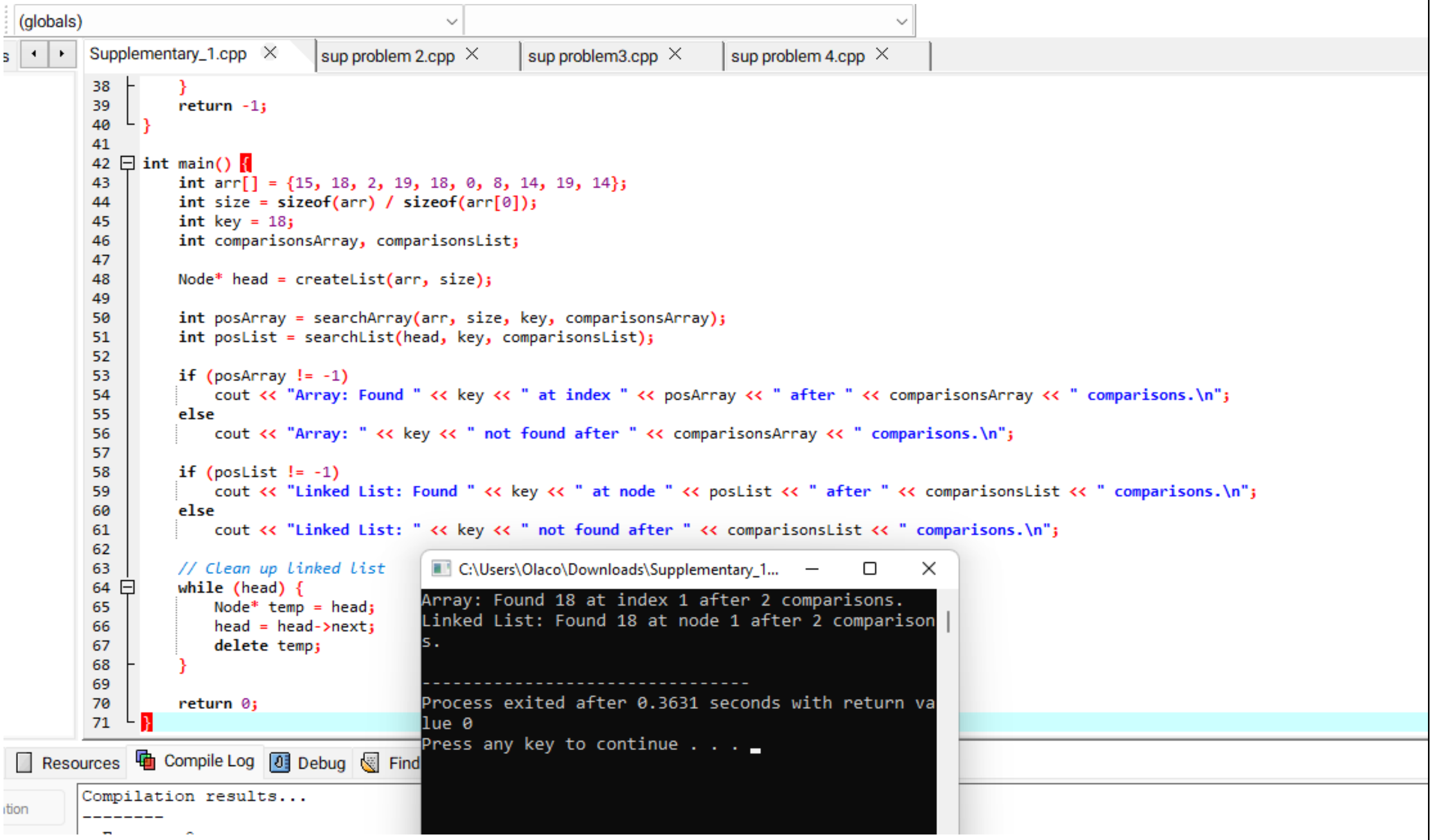
```cpp
#ifndef SEARCHING_H
#define SEARCHING_H

template <typename T>
Node<T>* binarySearchLinkedList(Node<T>* head, T key) {
    Node<T>* low = head;
    Node< >* high = nullptr;
    Node<T>* mid = nullptr;

    while (low != nullptr) {
        mid = low;
        high = low->next;

        // Check if mid is the correct match
        if (mid != nullptr && mid->data == key) {
            return mid;
        }

        if (mid->data < key) {
            low = mid->next;
        } else {
            high = mid;
        }
    }

    return nullptr; // Item not found
}

#endif
```



```cpp
#ifndef NODES_H
#define NODES_H

template <typename T>
struct Node {
    T data;
    Node* next;

    Node(T data) : data(data), next(nullptr) {}
};

template <typename T>
Node<T>* new_node(T data) {
    return new Node<T>(data);
}

#endif

```

7. Supplementary Activity

PROBLEM 1:



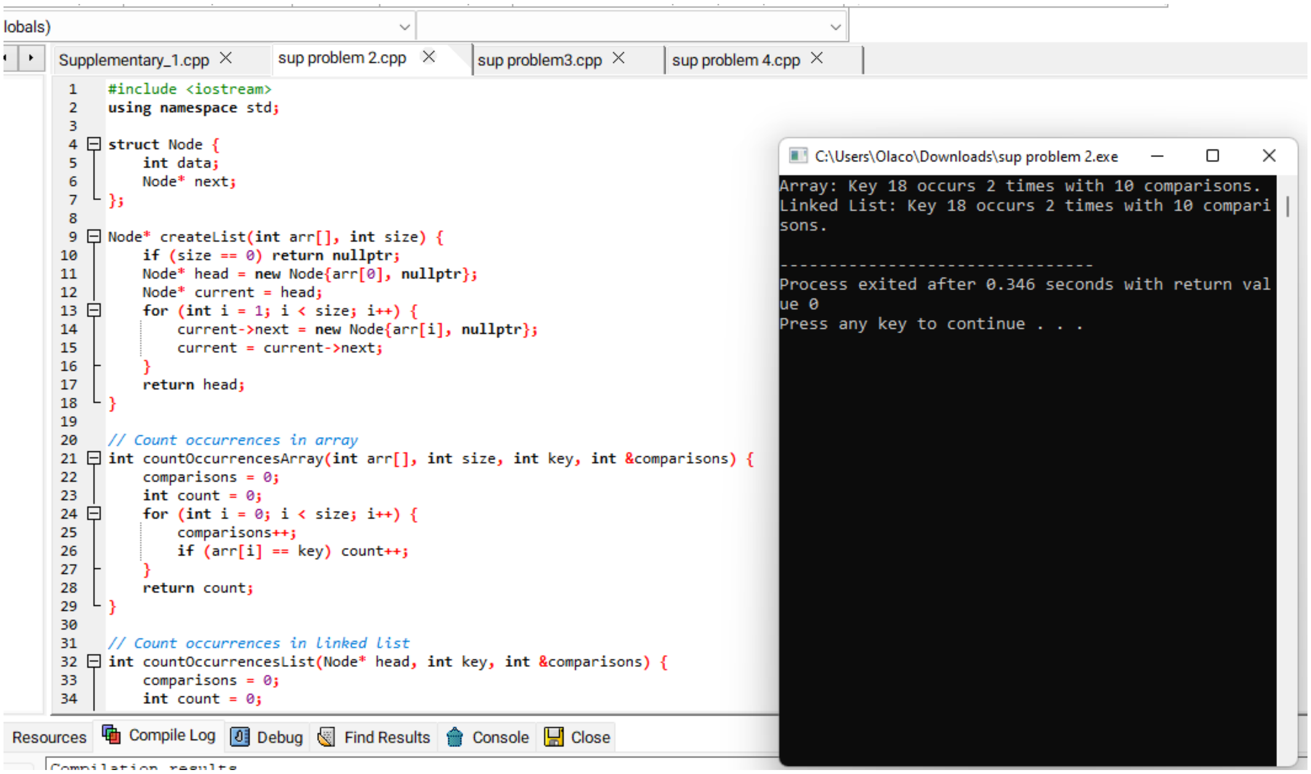
```cpp
    }
    return -1;
}

int main() {
    int arr[] = {15, 18, 2, 19, 18, 0, 8, 14, 19, 14};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key = 18;
    int comparisonsArray, comparisonsList;

    Node* head = createList(arr, size);

    int posArray = searchArray(arr, size, key, comparisonsArray);
    int posList = searchList(head, key, comparisonsList);

    if (posArray != -1)
        cout << "Array: Found " << key << " at index " << posArray << " after " << comparisonsArray << " comparisons.\n";
    else
        cout << "Array: " << key << " not found after " << comparisonsArray << " comparisons.\n";

    if (posList != -1)
        cout << "Linked List: Found " << key << " at node " << posList << " after " << comparisonsList << " comparisons.\n";
    else
        cout << "Linked List: " << key << " not found after " << comparisonsList << " comparisons.\n";

    // Clean up linked list
    while (head) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }

    return 0;
}
```

```
Array: Found 18 at index 1 after 2 comparisons.
Linked List: Found 18 at node 1 after 2 comparison
s.

--------------------------------
Process exited after 0.3631 seconds with return va
lue 0
Press any key to continue . . .
```

```
Compilation results...
--------
```

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:

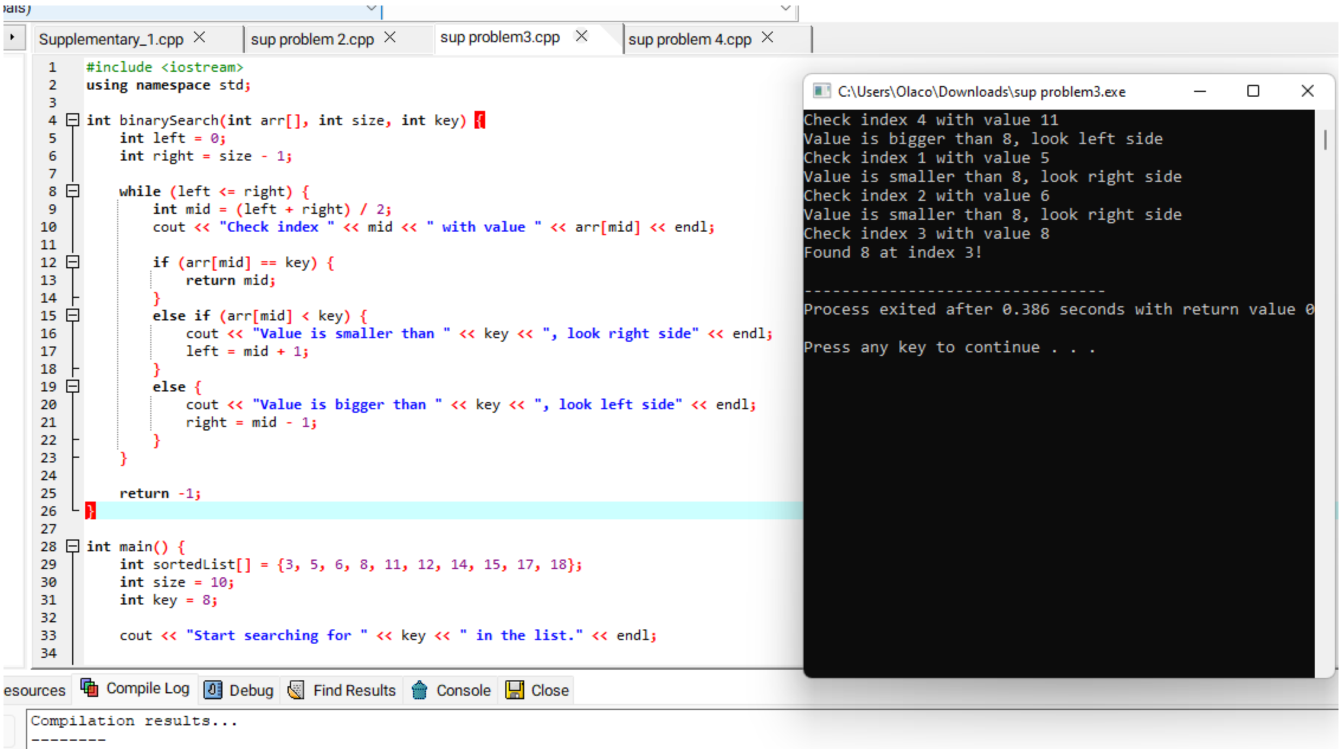


```cpp
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

Node* createList(int arr[], int size) {
    if (size == 0) return nullptr;
    Node* head = new Node{arr[0], nullptr};
    Node* current = head;
    for (int i = 1; i < size; i++) {
        current->next = new Node{arr[i], nullptr};
        current = current->next;
    }
    return head;
}

// Count occurrences in array
int countOccurrencesArray(int arr[], int size, int key, int &comparisons) {
    comparisons = 0;
    int count = 0;
    for (int i = 0; i < size; i++) {
        comparisons++;
        if (arr[i] == key) count++;
    }
    return count;
}

// Count occurrences in linked list
int countOccurrencesList(Node* head, int key, int &comparisons) {
    comparisons = 0;
    int count = 0;
```

```
Array: Key 18 occurs 2 times with 10 comparisons.
Linked List: Key 18 occurs 2 times with 10 compari
sons.

--------------------------------
Process exited after 0.346 seconds with return val
ue 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:



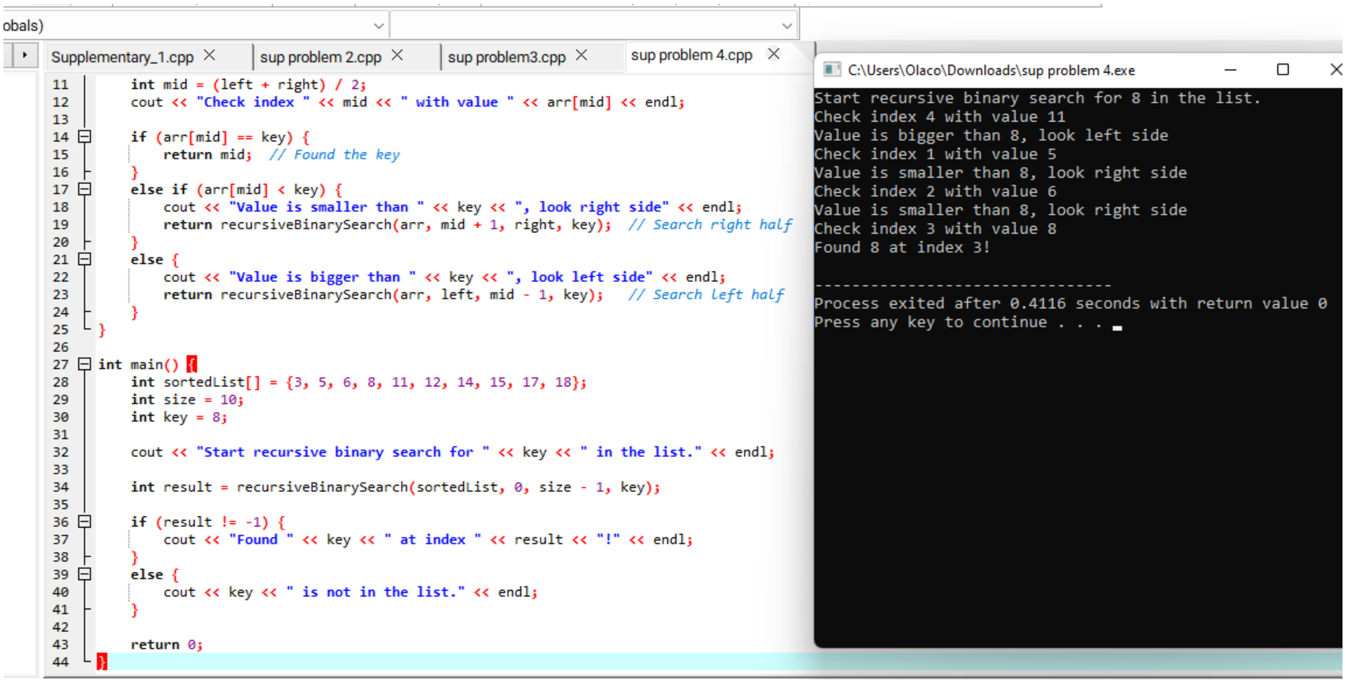
```cpp
#include <iostream>
using namespace std;

int binarySearch(int arr[], int size, int key) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = (left + right) / 2;
        cout << "Check index " << mid << " with value " << arr[mid] << endl;

        if (arr[mid] == key) {
            return mid;
        }
        else if (arr[mid] < key) {
            cout << "Value is smaller than " << key << ", look right side" << endl;
            left = mid + 1;
        }
        else {
            cout << "Value is bigger than " << key << ", look left side" << endl;
            right = mid - 1;
        }
    }

    return -1;
}

int main() {
    int sortedList[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
    int size = 10;
    int key = 8;

    cout << "Start searching for " << key << " in the list." << endl;
```

```
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 . . .
```

```
Compilation results...
--------
```

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:



```cpp
    int mid = (left + right) / 2;
    cout << "Check index " << mid << " with value " << arr[mid] << endl;

    if (arr[mid] == key) {
        return mid;  // Found the key
    }
    else if (arr[mid] < key) {
        cout << "Value is smaller than " << key << ", look right side" << endl;
        return recursiveBinarySearch(arr, mid + 1, right, key);  // Search right half
    }
    else {
        cout << "Value is bigger than " << key << ", look left side" << endl;
        return recursiveBinarySearch(arr, left, mid - 1, key);   // Search left half
    }
}

int main() {
    int sortedList[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
    int size = 10;
    int key = 8;

    cout << "Start recursive binary search for " << key << " in the list." << endl;

    int result = recursiveBinarySearch(sortedList, 0, size - 1, key);

    if (result != -1) {
        cout << "Found " << key << " at index " << result << "!" << endl;
    }
    else {
        cout << key << " is not in the list." << endl;
    }

    return 0;
}
```

```
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 seriatim 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