Activity No. 6.1	
Searching Techniques	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 10/15/24**
Section: CPE21S4	**Date Submitted: 10/16/24**
Name(s): SANTOS, ANDREI R.	**Instructor: Professor Maria Rizette Sayo**
6. Output	

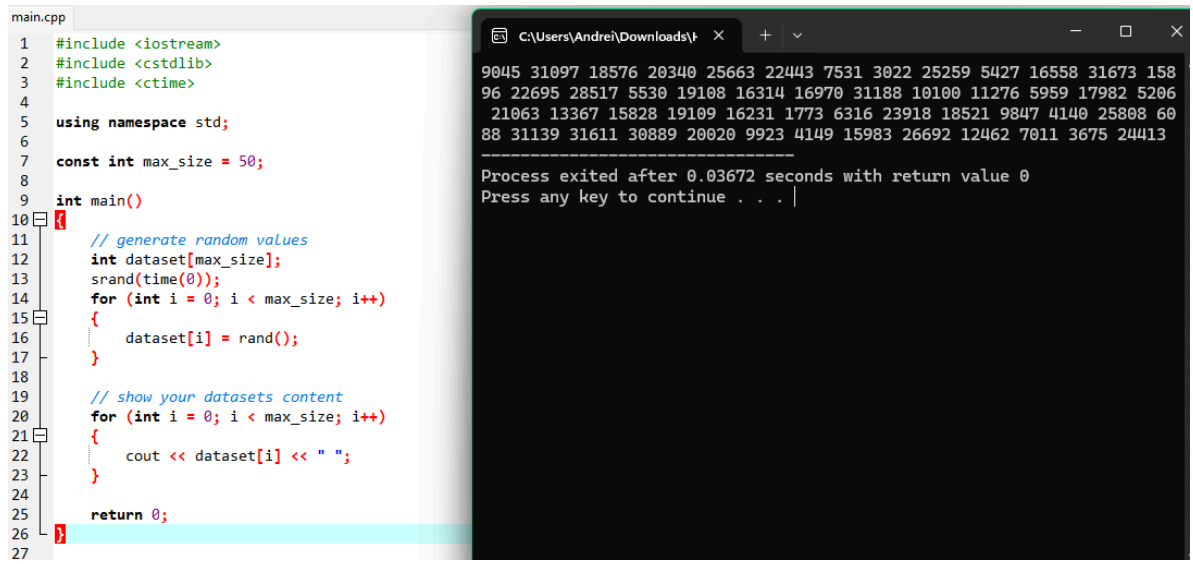
Screenshot	
Observations	The program runs and shows random values that are based on the number of the size given in the code.

Table 6-1. Data Generated and Observations.

Code	// main.cpp
	#include <iostream>
	#include "nodes.h"
	#include "searching.h"
	using namespace std;
	
	int main() {
	// Create a linked list: 12 -> 24 -> 36 -> 48 -> 60 -> 72
	Node<int>* head = new_node(12);
	head->next = new_node(24);
	head->next->next = new_node(36);
	head->next->next->next = new_node(48);
	head->next->next->next->next = new_node(60);
	head->next->next->next->next->next = new_node(72);
	
	// Prompt user for the item to search
	int itemToSearch;
	cout << "Enter the value to search: ";

	cin >> itemToSearch;
	
	// Perform linear search
	linearSearch(head, itemToSearch);
	
	Node<int>* current = head;
	while (current != NULL) {
	Node<int>* nextNode = current->next;
	delete current;
	current = nextNode;
	}
	
	return 0;
	}
	
	NOTE : using namespace was used.

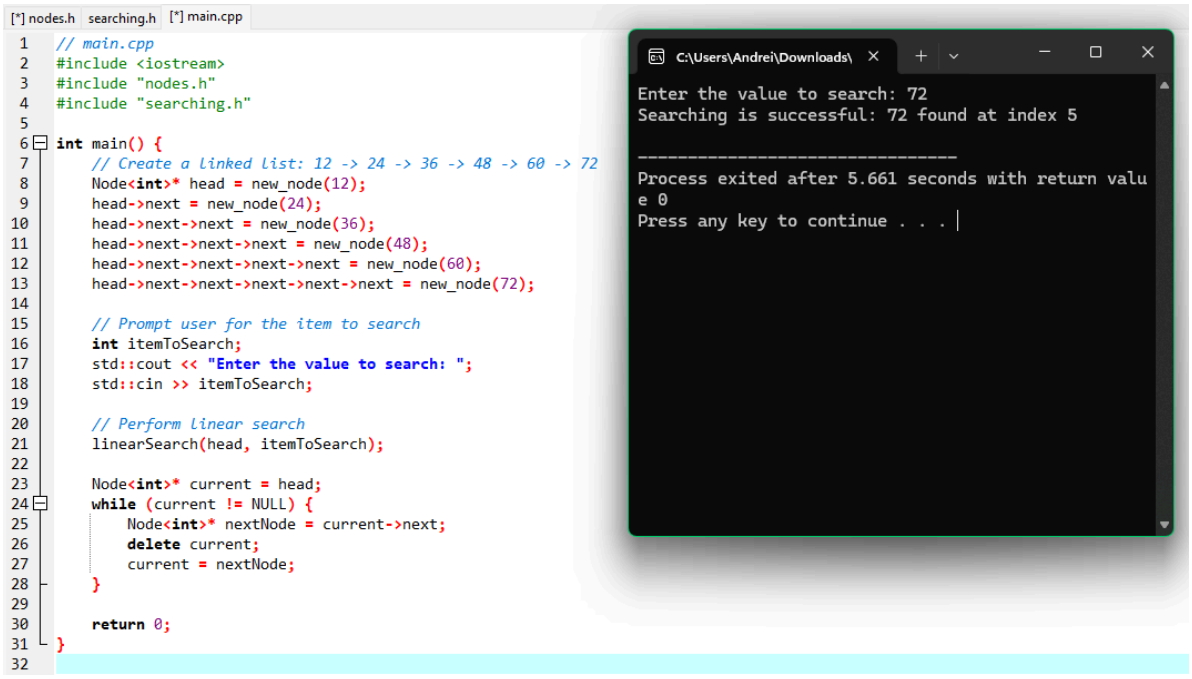
Output	
Observations	This program creates a linked list with the values 12, 24, 36, 48, 60, and 72. It prompts the user to enter a value to search in the linked list using a linear search function. After the search, it deletes all nodes in the linked list to free memory.

Table 6-2a. Linear Search for Arrays

Code	#include <iostream>
	#include "nodes.h"
	#include "searching.h"
	using namespace std;
	
	int main() {
	// Create linked list for the name "Andrei"

```
    Node<char>* name1 = new_node('A');
    Node<char>* name2 = new_node('n');
    Node<char>* name3 = new_node('d');
    Node<char>* name4 = new_node('r');
    Node<char>* name5 = new_node('e');
    Node<char>* name6 = new_node('i');

    // Link each node to each other
    name1->next = name2;
    name2->next = name3;
    name3->next = name4;
    name4->next = name5;
    name5->next = name6;
    name6->next = NULL;

    // letter to search
    char dataToFind;
    cout << "Enter a letter to search in the linked list: ";
    cin >> dataToFind;

    // linear search
    if (linearLS(name1, dataToFind)) {
        std::cout << "Letter '" << dataToFind << "' found in the linked list." << std::endl;
    } else {
        std::cout << "Letter '" << dataToFind << "' not found in the linked list." << std::endl;
    }

    return 0;
}
```

NOTE : using namespace was used.

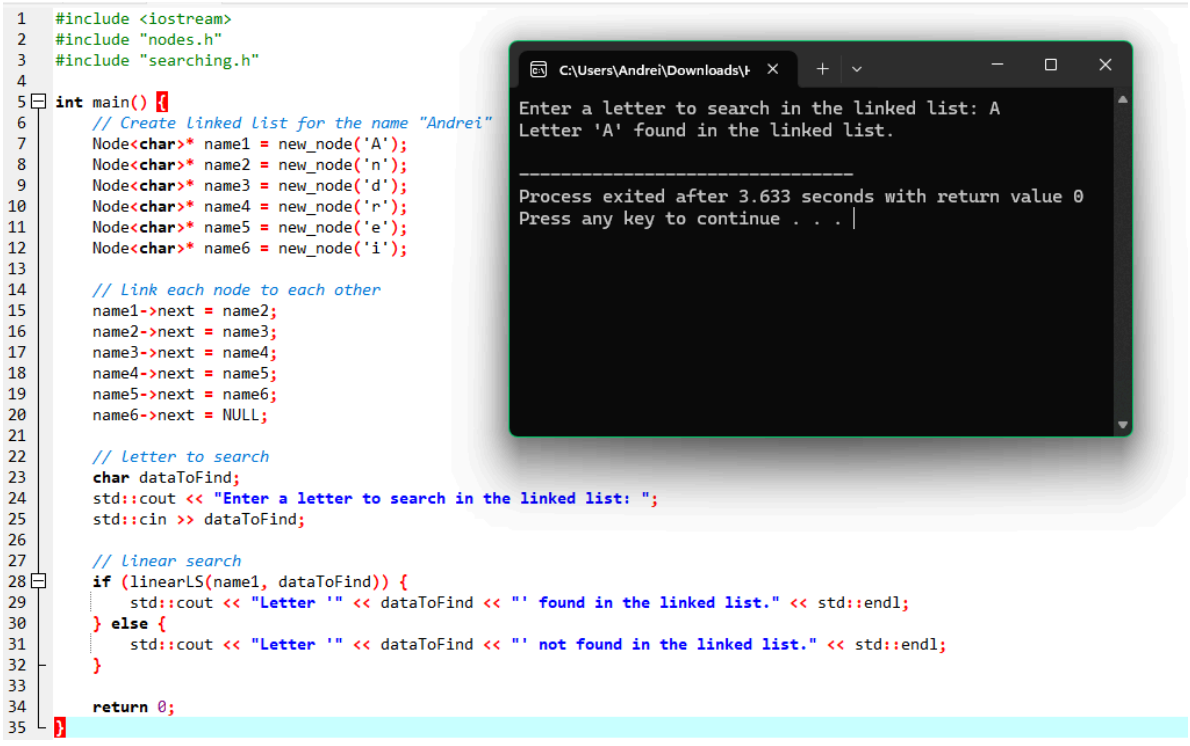
Output	
Observations	The program compiles a linked list representing the name "Andrei," with each character stored in separate nodes. It prompts the user to enter a letter to search for in the linked list using a linear search function. The program then checks if the letter exists in the list and outputs whether it was found or not.

Table 6-2b. Linear Search for Linked List

Code	#include <iostream>
	#include "searching.h"
	#include "nodes.h"
	using namespace std;
	
	int main() {
	
	int arr[] = {2, 5, 8, 12, 16, 24, 25, 36, 48, 56}; // Sorted array
	int n = sizeof(arr) / sizeof(arr[0]);
	int searchElement;
	
	cout << "Enter the element to search: ";
	cin >> searchElement;
	
	binarySearch(arr, n, searchElement);
	
	return 0;
	}

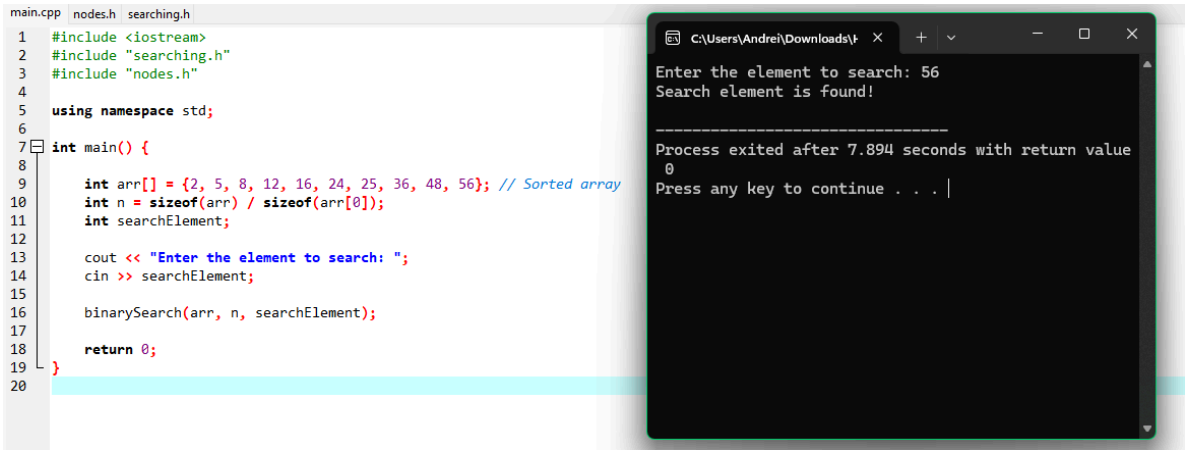
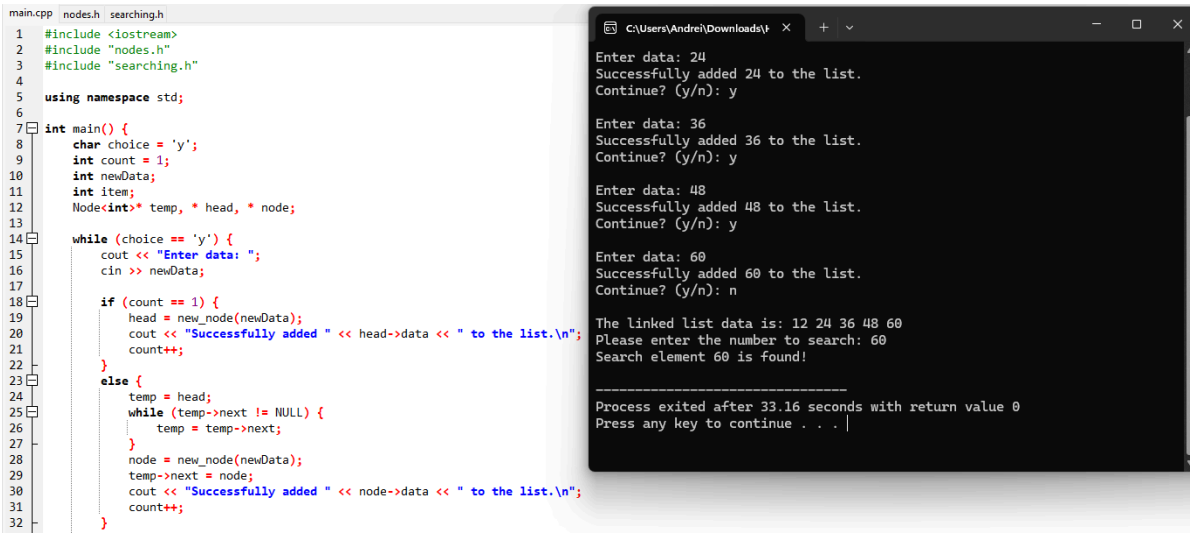
Output	
Observations	The code defines a sorted array of integers and prompts the user to enter an element to search for. It then performs a binary search on the array to find the specified element. The result of the search will indicate whether the element was found in the array or not.

Table 6-3a. Binary Search for Arrays

Code	#include <iostream>
	#include "nodes.h"
	#include "searching.h"
	
	using namespace std;
	
	int main() {
	  char choice = 'y';
	  int count = 1;
	  int newData;
	  int item;
	  Node<int>* temp, * head, * node;
	
	  while (choice == 'y') {
	  cout << "Enter data: ";
	  cin >> newData;
	
	  if (count == 1) {
	  head = new_node(newData);
	  cout << "Successfully added " << head->data << " to the list.\n";
	  count++;
	  }
	  else {
	  temp = head;
	  while (temp->next != NULL) {
	  temp = temp->next;
	  }

```
            node = new_node(newData);
            temp->next = node;
            cout << "Successfully added " << node->data << " to the list.\n";
            count++;
        }

        // Check if wants to continue
        cout << "Continue? (y/n): ";
        cin >> choice;
        cout << endl;
    }

    // Display the linked list
    cout << "The linked list data is: ";
    Node<int>* currNode = head;
    while (currNode != NULL) {
        cout << currNode->data << " ";
        currNode = currNode->next;
    }
    cout << endl;

    // Search for an element in the linked list
    cout << "Please enter the number to search: ";
    cin >> item;
    binarySearch(head, item);

    return 0;
}
```

Output



Observations

The program lets you build a linked list by inputting integers. You can keep adding numbers until you choose not to, and then it displays the entire list. Finally, it asks for a number to search for in the linked list using binary search.

Table 6-3b. Binary Search for Linked List

7. Supplementary Activity

PROBLEMS

1.

ARRAY

Supplementary Activity # 1 (Arr).cpp [*] Supplementary Activity # 1 (L.L).cpp

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

int sequentialSearch(int arr[], int size, int key) {
    int comparisons = 0;
    for (int i = 0; i < size; i++) {
        comparisons++;
        if (arr[i] == key) {
            cout << "Found key: " << key << " in the index of
            return comparisons; // return the number of compar
        }
    }
    return comparisons; // return comparisons even if the key
}

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

    cout << "Enter the key to search for: ";
    cin >> key; // User input for the key to search
    int size = sizeof(arr) / sizeof(arr[0]);

    int comparisons = sequentialSearch(arr, size, key);
    cout << "Number of comparisons in array: " << comparisons << endl;

    return 0;
}

```

```
C:\Users\Andrei\Downloads\S
Enter the key to search for: 18
Found key: 18 in the index of 1
Number of comparisons in array: 2

--------------------------------
Process exited after 1.518 seconds with return value 0
Press any key to continue . . .
```

LINKED LIST

Supplementary Activity # 1 (Arr).cpp Supplementary Activity # 1 (L.L).cpp

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

// Node structure for the linked list
struct Node {
    int data;
    Node* next;
};

// Function to create a new node
Node* newNode(int data) {
    Node* node = new Node();
    node->data = data;
    node->next = NULL;
    return node;
}

// Sequential search function for the linked list
int linkedListSearch(Node* head, int key) {
    int comparisons = 0;
    Node* current = head;
    while (current != NULL) {
        comparisons++;
        if (current->data == key) {
            cout << "Found key: " << key << " in the linked list" << endl; // Display the found key
            return comparisons; // return the number of comparisons made
        }
```

```
C:\Users\Andrei\Downloads\S
Enter the key to search for: 18
Found key: 18 in the linked list
Number of comparisons in linked list: 2

--------------------------------
Process exited after 0.7963 seconds with return value 0
Press any key to continue . . .
```

2.

ARRAY

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

int countRepeatsInArray(int arr[], int size, int key) {
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            count++;
        }
    }
    return count;
}

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

    cout << "Enter the key to search for: ";
    cin >> key;
    int count = countRepeatsInArray(arr, size, key);
    cout << "Number of repeating instances of " << key << " in array: " << count << endl;

    return 0;
}

```

input

```
Enter the key to search for: 18
Number of repeating instances of 18 in array: 2


...Program finished with exit code 0
Press ENTER to exit console.
```

LINKED LIST

main.cpp

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

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

Node* newNode(int data) {
    Node* node = new Node();
    node->data = data;
    node->next = NULL;
    return node;
}

int countRepeatsInList(Node* head, int key) {
    int count = 0;
    Node* current = head;
    while (current != NULL) {
        if (current->data == key) {
            count++;
        }
        current = current->next;
    }
    return count;
}

int main() {
    Node* head = newNode(15);
    head->next = newNode(18);
    head->next->next = newNode(2);
    head->next->next->next = newNode(19);
    head->next->next->next->next = newNode(18);
    head->next->next->next->next->next = newNode(0);
    head->next->next->next->next->next->next = newNode(8);
    head->next->next->next->next->next->next->next = newNode(14);
    head->next->next->next->next->next->next->next->next = newNode(19);
```

```
Enter the key to search for: 18
Number of repeating instances of 18 in linked list: 2


...Program finished with exit code 0
Press ENTER to exit console.
```

3.

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


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

    while (left <= right) {
        int mid = left + (right - left) / 2;


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


        if (arr[mid] < key) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1; // Key not found
}

int main() {
    int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18}; // Sorted array
    int size = sizeof(arr) / sizeof(arr[0]); // Calculate size
    int key = 8;

    int result = binarySearch(arr, size, key);
    if (result != -1) {
        cout << "Element " << key << " found at index: " << result << endl; // Show result
    } else {
        cout << "Element " << key << " not found." << endl;
    }
```

input

```
Element 8 found at index: 3
```

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


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

    while (left <= right) {
        int mid = left + (right - left) / 2;


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


        if (arr[mid] < key) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return -1; // Key not found
}

int main() {
    int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18}; // Sorted arra
    int size = sizeof(arr) / sizeof(arr[0]); // Calculate size
    int key = 8;

    int result = binarySearch(arr, size, key);
    if (result != -1) {
        cout << "Element " << key << " found at index: " << result <<
    } else {
        cout << "Element " << key << " not found." << endl;
    }
```

ITERATION
L => R

FOUND AT
INDEX 3

```
lement 8 found at index: 3


..Program finished with exit code 0
ress ENTER to exit console.
```

4.

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

int recursiveBinarySearch(int arr[], int low, int up, int key) {
    if (low > up) {
        return -1;
    }

    int mid = low + (up - low) / 2;

    if (arr[mid] == key) {
        return mid;
    }
    else if (arr[mid] < key) {
        return recursiveBinarySearch(arr, mid + 1, up, key);
    }
    else {
        return recursiveBinarySearch(arr, low, mid - 1, key);
    }
}

int main() {
    int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
    int size = sizeof(arr) / sizeof(arr[0]);
    int key = 8;

    int result = recursiveBinarySearch(arr, 0, size - 1, key);
    if (result != -1) {
        cout << "Element found at index: " << result << endl;
    } else {
        cout << "Element not found!" << endl;
    }

    return 0;
}
```

```
Element found at index: 3


...Program finished with exit code 0
Press ENTER to exit console.
```

8. Conclusion

In this activity, I learned how to create and manage linked lists in C++, focusing on dynamic insertion and searching. The procedure involved building a linked list by taking user input and implementing search functionality, which helped me understand linked list operations better. The supplementary activity emphasized the limitations of binary search with linked lists, as I had to use linear search instead, reinforcing the need for appropriate algorithms based on data structures. Overall, I believe I performed well in this activity by successfully implementing the required functions; however, I recognize the need to improve my understanding of optimizing search operations and recursion in more complex data structures.

9. Assessment Rubric