

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	<pre> main.cpp 1  #include &lt;iostream&gt; 2  #include &lt;cstdlib&gt; 3  #include &lt;ctime&gt; 4 5  using namespace std; 6 7  const int max_size = 50; 8 9  int main() 10 { 11     // generate random values 12     int dataset[max_size]; 13     srand(time(0)); 14     for (int i = 0; i &lt; max_size; i++) 15     { 16         dataset[i] = rand(); 17     } 18 19     // show your datasets content 20     for (int i = 0; i &lt; max_size; i++) 21     { 22         cout &lt;&lt; dataset[i] &lt;&lt; " "; 23     } 24 25     return 0; 26 } 27 C:\Users\Andre\Downloads\... 9045 31097 18576 20340 25663 22443 7531 3022 25259 5427 16558 31673 158 96 22695 28517 5530 19108 16314 16970 31188 10100 11276 5959 17982 5206 21063 13367 15828 19109 16231 1773 6316 23918 18521 9847 4140 25808 60 88 31139 31611 30889 20020 9923 4149 15983 26692 12462 7011 3675 24413 ----- Process exited after 0.03672 seconds with return value 0 Press any key to continue . . . </pre>
Observations	<p>The program runs and shows random values that are based on the number of the size given in the code.</p>

Table 6-1. Data Generated and Observations.

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

	<pre> cin &gt;&gt; itemToSearch;  // Perform linear search linearSearch(head, itemToSearch);  Node&lt;int&gt;* current = head; while (current != NULL) {     Node&lt;int&gt;* nextNode = current-&gt;next;     delete current;     current = nextNode; }  return 0; } </pre> <p><b>NOTE : using namespace was used.</b></p>
Output	<div> <div> <pre> [?] nodes.h  searching.h  [?] main.cpp 1 // main.cpp 2 #include &lt;iostream&gt; 3 #include "nodes.h" 4 #include "searching.h" 5 6 int main() { 7     // Create a Linked List: 12 -&gt; 24 -&gt; 36 -&gt; 48 -&gt; 60 -&gt; 72 8     Node&lt;int&gt;* head = new_node(12); 9     head-&gt;next = new_node(24); 10    head-&gt;next-&gt;next = new_node(36); 11    head-&gt;next-&gt;next-&gt;next = new_node(48); 12    head-&gt;next-&gt;next-&gt;next-&gt;next = new_node(60); 13    head-&gt;next-&gt;next-&gt;next-&gt;next-&gt;next = new_node(72); 14 15    // Prompt user for the item to search 16    int itemToSearch; 17    std::cout &lt;&lt; "Enter the value to search: "; 18    std::cin &gt;&gt; itemToSearch; 19 20    // Perform Linear search 21    linearSearch(head, itemToSearch); 22 23    Node&lt;int&gt;* current = head; 24    while (current != NULL) { 25        Node&lt;int&gt;* nextNode = current-&gt;next; 26        delete current; 27        current = nextNode; 28    } 29 30    return 0; 31 } 32 </pre> </div> <div> </div> </div>
Observations	<p>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.</p>

Table 6-2a. Linear Search for Arrays

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

```
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.**

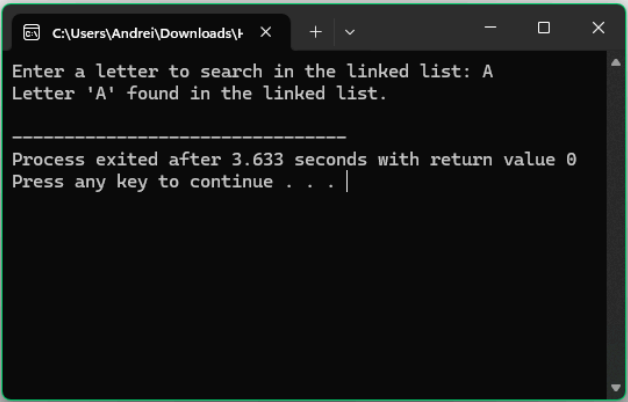
Output	<pre> 1  #include &lt;iostream&gt; 2  #include "nodes.h" 3  #include "searching.h" 4 5  int main() { 6      // Create Linked List for the name "Andrei" 7      Node&lt;char*&gt; name1 = new_node('A'); 8      Node&lt;char*&gt; name2 = new_node('n'); 9      Node&lt;char*&gt; name3 = new_node('d'); 10     Node&lt;char*&gt; name4 = new_node('r'); 11     Node&lt;char*&gt; name5 = new_node('e'); 12     Node&lt;char*&gt; name6 = new_node('i'); 13 14     // Link each node to each other 15     name1-&gt;next = name2; 16     name2-&gt;next = name3; 17     name3-&gt;next = name4; 18     name4-&gt;next = name5; 19     name5-&gt;next = name6; 20     name6-&gt;next = NULL; 21 22     // Letter to search 23     char dataToFind; 24     std::cout &lt;&lt; "Enter a letter to search in the linked list: "; 25     std::cin &gt;&gt; dataToFind; 26 27     // Linear search 28     if (linearLS(name1, dataToFind)) { 29         std::cout &lt;&lt; "Letter '" &lt;&lt; dataToFind &lt;&lt; "' found in the linked list." &lt;&lt; std::endl; 30     } else { 31         std::cout &lt;&lt; "Letter '" &lt;&lt; dataToFind &lt;&lt; "' not found in the linked list." &lt;&lt; std::endl; 32     } 33 34     return 0; 35 } </pre> 
Observations	<p>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.</p>

Table 6-2b. Linear Search for Linked List

Code	<pre> #include &lt;iostream&gt; #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 &lt;&lt; "Enter the element to search: ";     cin &gt;&gt; searchElement;      binarySearch(arr, n, searchElement);      return 0; } </pre>
------	---

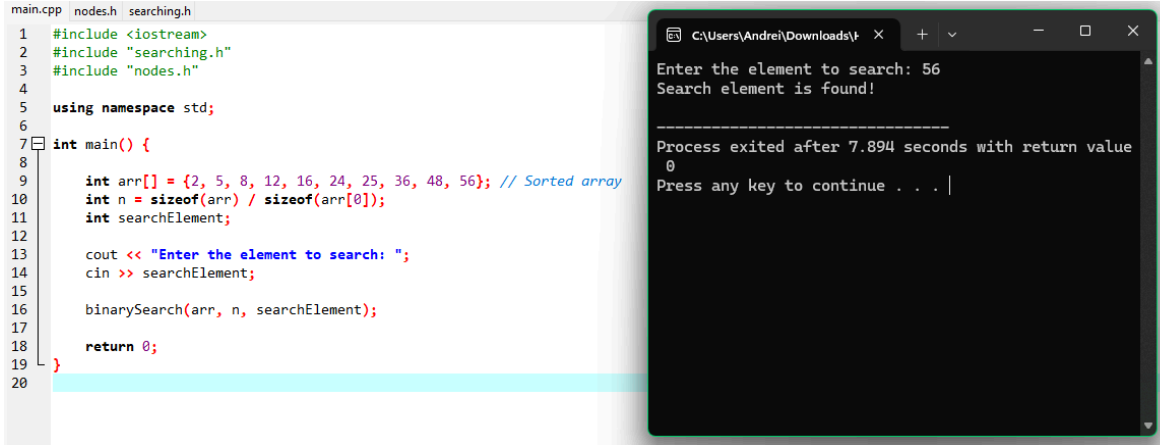
Output	 <pre> main.cpp  nodes.h  searching.h 1  #include &lt;iostream&gt; 2  #include "searching.h" 3  #include "nodes.h" 4 5  using namespace std; 6 7  int main() { 8 9      int arr[] = {2, 5, 8, 12, 16, 24, 25, 36, 48, 56}; // Sorted array 10     int n = sizeof(arr) / sizeof(arr[0]); 11     int searchElement; 12 13     cout &lt;&lt; "Enter the element to search: "; 14     cin &gt;&gt; searchElement; 15 16     binarySearch(arr, n, searchElement); 17 18     return 0; 19 } 20 Enter the element to search: 56 Search element is found!  ----- Process exited after 7.894 seconds with return value 0 Press any key to continue . . .   </pre>
Observations	<p>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.</p>

Table 6-3a. Binary Search for Arrays

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

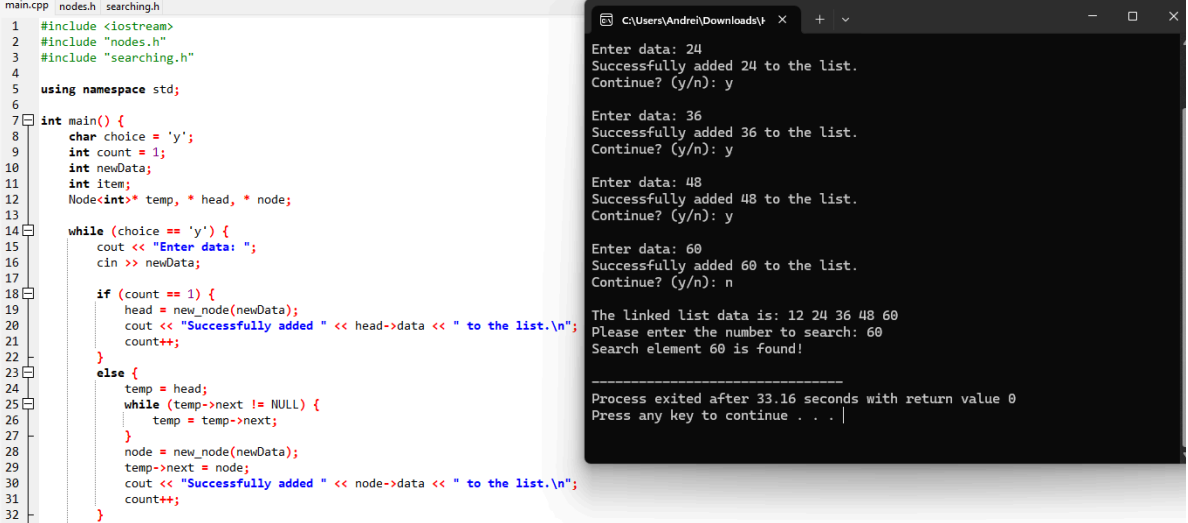
	<pre>         node = new_node(newData);         temp-&gt;next = node;         cout &lt;&lt; "Successfully added " &lt;&lt; node-&gt;data &lt;&lt; " to the list.\n";         count++;     }      // Check if wants to continue     cout &lt;&lt; "Continue? (y/n): ";     cin &gt;&gt; choice;     cout &lt;&lt; endl; }  // Display the linked list cout &lt;&lt; "The linked list data is: "; Node&lt;int&gt;* currNode = head; while (currNode != NULL) {     cout &lt;&lt; currNode-&gt;data &lt;&lt; " ";     currNode = currNode-&gt;next; } cout &lt;&lt; endl;  // Search for an element in the linked list cout &lt;&lt; "Please enter the number to search: "; cin &gt;&gt; item; binarySearch(head, item);  return 0; } </pre>
Output	 <pre> main.cpp  nodes.h  searching.h 1  #include &lt;iostream&gt; 2  #include "nodes.h" 3  #include "searching.h" 4 5  using namespace std; 6 7  int main() { 8      char choice = 'y'; 9      int count = 1; 10     int newData; 11     int item; 12     Node&lt;int&gt;* temp, * head, * node; 13 14     while (choice == 'y') { 15         cout &lt;&lt; "Enter data: "; 16         cin &gt;&gt; newData; 17 18         if (count == 1) { 19             head = new_node(newData); 20             cout &lt;&lt; "Successfully added " &lt;&lt; head-&gt;data &lt;&lt; " to the list.\n"; 21             count++; 22         } 23         else { 24             temp = head; 25             while (temp-&gt;next != NULL) { 26                 temp = temp-&gt;next; 27             } 28             node = new_node(newData); 29             temp-&gt;next = node; 30             cout &lt;&lt; "Successfully added " &lt;&lt; node-&gt;data &lt;&lt; " to the list.\n"; 31             count++; 32         } 33     } 34 35     // Display the linked list 36     cout &lt;&lt; "The linked list data is: "; 37     Node&lt;int&gt;* currNode = head; 38     while (currNode != NULL) { 39         cout &lt;&lt; currNode-&gt;data &lt;&lt; " "; 40         currNode = currNode-&gt;next; 41     } 42     cout &lt;&lt; endl; 43 44     // Search for an element in the linked list 45     cout &lt;&lt; "Please enter the number to search: "; 46     cin &gt;&gt; item; 47     binarySearch(head, item); 48 49     return 0; 50 } </pre> <pre> Enter data: 24 Successfully added 24 to the list. Continue? (y/n): y  Enter data: 36 Successfully added 36 to the list. Continue? (y/n): y  Enter data: 48 Successfully added 48 to the list. Continue? (y/n): y  Enter data: 60 Successfully added 60 to the list. Continue? (y/n): n  The linked list data is: 12 24 36 48 60 Please enter the number to search: 60 Search element 60 is found!  Process exited after 33.16 seconds with return value 0 Press any key to continue . . . </pre>
Observations	<p>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.</p>

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
1  #include <iostream>
2  using namespace std;
3
4  int sequentialSearch(int arr[], int size, int key) {
5      int comparisons = 0;
6      for (int i = 0; i < size; i++) {
7          comparisons++;
8          if (arr[i] == key) {
9              cout << "Found key: " << key << " in the index of ";
10             return comparisons; // return the number of comparisons
11         }
12     }
13     return comparisons; // return comparisons even if the key is not found
14 }
15
16 int main() {
17     int arr[] = {15, 18, 2, 19, 18, 0, 8, 14, 19, 14};
18     int key;
19
20     cout << "Enter the key to search for: ";
21     cin >> key; // User input for the key to search
22     int size = sizeof(arr) / sizeof(arr[0]);
23
24     int comparisons = sequentialSearch(arr, size, key);
25     cout << "Number of comparisons in array: " << comparisons << endl;
26
27     return 0;
28 }
29
```

```
C:\Users\Andrei\Downloads\S$ X + v
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
1  #include <iostream>
2  using namespace std;
3
4  // Node structure for the linked list
5  struct Node {
6      int data;
7      Node* next;
8  };
9
10 // Function to create a new node
11 Node* newNode(int data) {
12     Node* node = new Node();
13     node->data = data;
14     node->next = NULL;
15     return node;
16 }
17
18 // Sequential search function for the linked list
19 int linkedListSearch(Node* head, int key) {
20     int comparisons = 0;
21     Node* current = head;
22     while (current != NULL) {
23         comparisons++;
24         if (current->data == key) {
25             cout << "Found key: " << key << " in the linked list" << endl; // Display the found key
26             return comparisons; // return the number of comparisons made
27         }
28     }
29 }
```

```
C:\Users\Andrei\Downloads\S$ X + v
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

```
1  #include <iostream>
2  using namespace std;
3
4  int countRepeatsInArray(int arr[], int size, int key) {
5      int count = 0;
6      for (int i = 0; i < size; i++) {
7          if (arr[i] == key) {
8              count++;
9          }
10     }
11     return count;
12 }
13
14 int main() {
15     int arr[] = {15, 18, 2, 19, 18, 0, 8, 14, 19, 14};
16     int size = sizeof(arr) / sizeof(arr[0]);
17     int key;
18
19     cout << "Enter the key to search for: ";
20     cin >> key;
21     int count = countRepeatsInArray(arr, size, key);
22     cout << "Number of repeating instances of " << key << " in array: " << count << endl;
23
24     return 0;
25 }
26
```

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

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node {
5      int data;
6      Node* next;
7  };
8
9  Node* newNode(int data) {
10     Node* node = new Node();
11     node->data = data;
12     node->next = NULL;
13     return node;
14 }
15
16 int countRepeatsInList(Node* head, int key) {
17     int count = 0;
18     Node* current = head;
19     while (current != NULL) {
20         if (current->data == key) {
21             count++;
22         }
23         current = current->next;
24     }
25     return count;
26 }
27
28 int main() {
29     Node* head = newNode(15);
30     head->next = newNode(18);
31     head->next->next = newNode(2);
32     head->next->next->next = newNode(19);
33     head->next->next->next->next = newNode(18);
34     head->next->next->next->next->next = newNode(0);
35     head->next->next->next->next->next->next = newNode(8);
36     head->next->next->next->next->next->next->next = newNode(14);
37     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.

```
1  #include <iostream>
2  using namespace std;
3
4
5  int binarySearch(int arr[], int size, int key) {
6      int left = 0; // Starting index
7      int right = size - 1; // Ending index
8
9      while (left <= right) {
10         int mid = left + (right - left) / 2;
11
12
13         if (arr[mid] == key) {
14             return mid;
15         }
16
17
18         if (arr[mid] < key) {
19             left = mid + 1;
20         } else {
21             right = mid - 1;
22         }
23     }
24     return -1; // Key not found
25 }
26
27 int main() {
28     int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18}; // Sorted array
29     int size = sizeof(arr) / sizeof(arr[0]); // Calculate size
30     int key = 8;
31
32     int result = binarySearch(arr, size, key);
33     if (result != -1) {
34         cout << "Element " << key << " found at index: " << result << endl; // Show result
35     } else {
36         cout << "Element " << key << " not found." << endl;
37     }
}
```

input

Element 8 found at index: 3

```
#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;
    } else {
        cout << "Element " << key << " not found." << endl;
    }
}
```

Element 8 found at index: 3

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

ITERATION  
L  $\Rightarrow$  R

FOUND AT  
INDEX 3

4.

```
1  #include <iostream>
2  using namespace std;
3
4  int recursiveBinarySearch(int arr[], int low, int up, int key) {
5      if (low > up) {
6          return -1;
7      }
8
9      int mid = low + (up - low) / 2;
10
11     if (arr[mid] == key) {
12         return mid;
13     }
14     else if (arr[mid] < key) {
15         return recursiveBinarySearch(arr, mid + 1, up, key);
16     }
17     else {
18         return recursiveBinarySearch(arr, low, mid - 1, key);
19     }
20 }
21
22 int main() {
23     int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
24     int size = sizeof(arr) / sizeof(arr[0]);
25     int key = 8;
26
27     int result = recursiveBinarySearch(arr, 0, size - 1, key);
28     if (result != -1) {
29         cout << "Element found at index: " << result << endl;
30     } else {
31         cout << "Element not found!" << endl;
32     }
33
34     return 0;
35 }
36
```

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