| Activity No. 6 | |
|---|---|
| **SEARCHING TECHNIQUES** | |
| **Course Code:** CPE010 | **Program:** Computer Engineering |
| **Course Title:** Data Structures and Algorithms | **Date Performed:** 9/16/25 |
| **Section:** CPE21S4 | **Date Submitted:** 9/16/25 |
| **Name(s):** Bautista, Mariela S. | **Instructor:** Engr. Quejado |

**A. Procedure: Output(s) and Observation(s)**

**B. Supplementary Activity: Output(s) and Observation(s)**

For each provided problem, give a screenshot of your code, the output console, and your answers to the questions.

- Problem 1. Suppose you are doing a sequential search of the list [15, 18, 2, 19, 18, 0, 8, 14, 19, 14]. Utilizing both a linked list and an array approach to the list, use sequential search and identify how many comparisons would be necessary to find the key '18'?

Code

main.cpp

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

int seqSearchArray(int arr[], int n, int key) {
    int comparisons = 0;
    for (int i = 0; i < n; i++) {
        comparisons++;
        if (arr[i] == key) {
            cout << "Found " << key << " at index " << i << endl;
            cout << "Comparisons: " << comparisons << endl;
            return i;
        }
    }
    cout << key << " not found\n";
    return -1;
}

// Linked list node
struct Node {
    int data;
    Node* next;
};

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

// Linked list sequential search
int seqSearchLL(Node* head, int key) {
    int comparisons = 0;
    Node* temp = head;
    int pos = 0;
    while (temp != NULL) {
        comparisons++;
        if (temp->data == key) {
```

```
        if (temp->data == key) {
            cout << "Found " << key << " at position " << pos << endl;
            cout << "Comparisons: " << comparisons << endl;
            return pos;
        }
        temp = temp->next;
        pos++;
    }
    cout << key << " not found\n";
    return -1;
}

int main() {
    int arr[] = {15, 18, 2, 19, 18, 0, 8, 14, 19, 14};
    int n = 10;

    cout << "Array Sequential Search:\n";
    seqSearchArray(arr, n, 18);

    // Build linked list
    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);
    head->next->next->next->next->next->next->next->next->next = newNode(14);

    cout << "\nLinked List Sequential Search:\n";
    seqSearchLL(head, 18);

    return 0;
}
```

Output

```
Array Sequential Search:
Found 18 at index 1
Comparisons: 2

Linked List Sequential Search:
Found 18 at position 1
Comparisons: 2


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

- Problem 2. Modify your sequential search algorithm so that it returns the count of repeating instances for a given search element ‘k’. Test on the same list given in problem 1.

Code

main.cpp

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

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

int main() {
    int arr[] = {15, 18, 2, 19, 18, 0, 8, 14, 19, 14};
    int n = 10;

    int key = 18;
    int count = seqSearchCount(arr, n, key);
    cout << "Element " << key << " appears " << count << " times.\n";
    return 0;
}

```

| | |
|---|---|
| Output | Element 18 appears 2 times.<br><br>...Program finished with exit code 0<br>Press ENTER to exit console. |

- Problem 3. Suppose you have the following sorted list [3, 5, 6, 8, 11, 12, 14, 15, 17, 18] and are using the binary search algorithm. If you wanted to find the key 8, draw a diagram that shows how the searching works per iteration of the algorithm. Prove that your drawing is correct by implementing the algorithm and showing a screenshot of the code and the output console.

Code

main.cpp

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

int binarySearch(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    int step = 1;

    while (low <= high) {
        int mid = (low + high) / 2;
        cout << "Step " << step++ << ": mid=" << mid << " arr[mid]=" << arr[mid] << endl;

        if (arr[mid] == key) {
            cout << "Found " << key << " at index " << mid << endl;
            return mid;
        }
        else if (arr[mid] > key) high = mid - 1;
        else low = mid + 1;
    }
    cout << key << " not found\n";
    return -1;
}

int main() {
    int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
    int n = 10;
    binarySearch(arr, n, 8);
    return 0;
}
```

Output

```
Step 1: mid=4 arr[mid]=11
Step 2: mid=1 arr[mid]=5
Step 3: mid=2 arr[mid]=6
Step 4: mid=3 arr[mid]=8
Found 8 at index 3


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

- Problem 4. Modify the binary search algorithm so that the algorithm becomes recursive. Using this new recursive binary search, implement a solution to the same problem for problem 3

| | |
|---|---|
| Code | main.cpp<br>1 #include <iostream><br>2 using namespace std;<br>3<br>4 int recursiveBinarySearch(int arr[], int low, int high, int key, int step=1) {<br>5 if (low > high) return -1;<br>6<br>7 int mid = (low + high) / 2;<br>8 cout << "Step " << step << ": mid=" << mid << " arr[mid]=" << arr[mid] << endl;<br>9<br>10 if (arr[mid] == key) return mid;<br>11 else if (arr[mid] > key) return recursiveBinarySearch(arr, low, mid - 1, key, step+1);<br>12 else return recursiveBinarySearch(arr, mid + 1, high, key, step+1);<br>13 }<br>14<br>15 int main() {<br>16 int arr[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};<br>17 int n = 10;<br>18 int result = recursiveBinarySearch(arr, 0, n - 1, 8);<br>19 if (result != -1) cout << "Found 8 at index " << result << endl;<br>20 else cout << "Not found\n";<br>21 return 0;<br>22 }<br>23 |
| Output | Step 1: mid=4 arr[mid]=11<br>Step 2: mid=1 arr[mid]=5<br>Step 3: mid=2 arr[mid]=6<br>Step 4: mid=3 arr[mid]=8<br>Found 8 at index 3<br><br>...Program finished with exit code 0<br>Press ENTER to exit console. |

x

**C. Conclusion & Lessons Learned**

In this activity, I learned the difference between sequential and binary search and how their efficiency changes depending on whether the data is stored in arrays or linked lists. Coding both versions helped me see that sequential search is simpler but slower, while binary search works faster in arrays but loses some of its advantage in linked lists since finding the middle node still requires traversal. The supplementary tasks, like counting duplicates and making the binary search recursive, pushed me to really understand the flow instead of just copying code.

I think I did fairly well since I was able to implement and test the algorithms correctly, though I noticed I still need to work on writing cleaner code on the first try instead of fixing errors after compiling because this is where I struggled the most. The procedure gave me a clearer picture of how time complexity plays out in actual programs, and the extra activity trained me to adjust algorithms for different requirements. If I improve my coding discipline, I'll be able to handle future activities more smoothly.