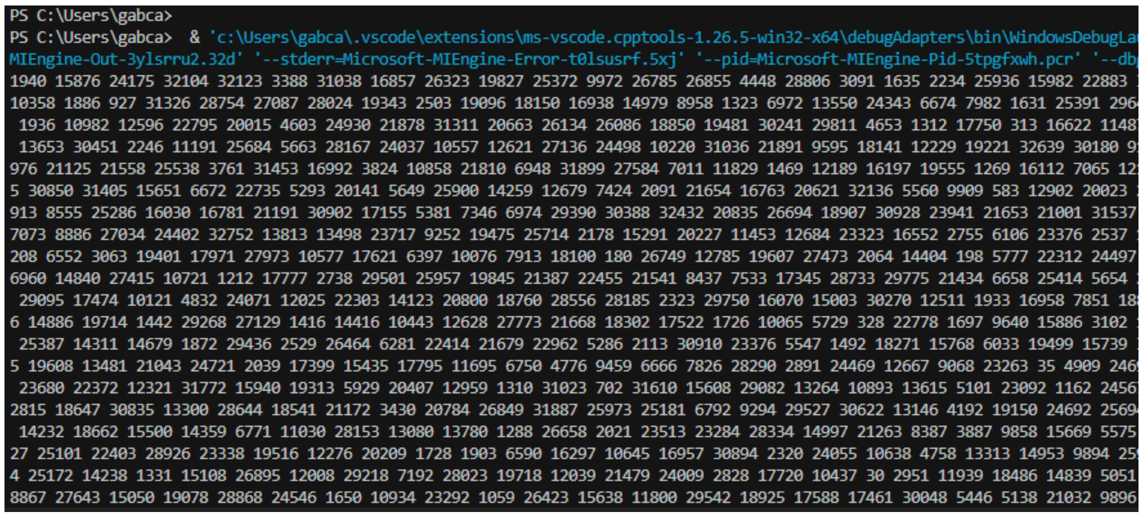
| | |
|---|---|
| **Hands-on Activity 6.1** | |
| **Searching Techniques** | |
| **Course Code:** CPE010 | **Program:** Computer Engineering |
| **Course Title:** Data Structures and Algorithms | **Date Performed:** 09/16/2025 |
| **Section:** CPE 21S4 | **Date Submitted:** 09/16/2025 |
| **Name(s):** Cabrera, Gabriel A. | **Instructor:** Engr. Jimlord Quejado |

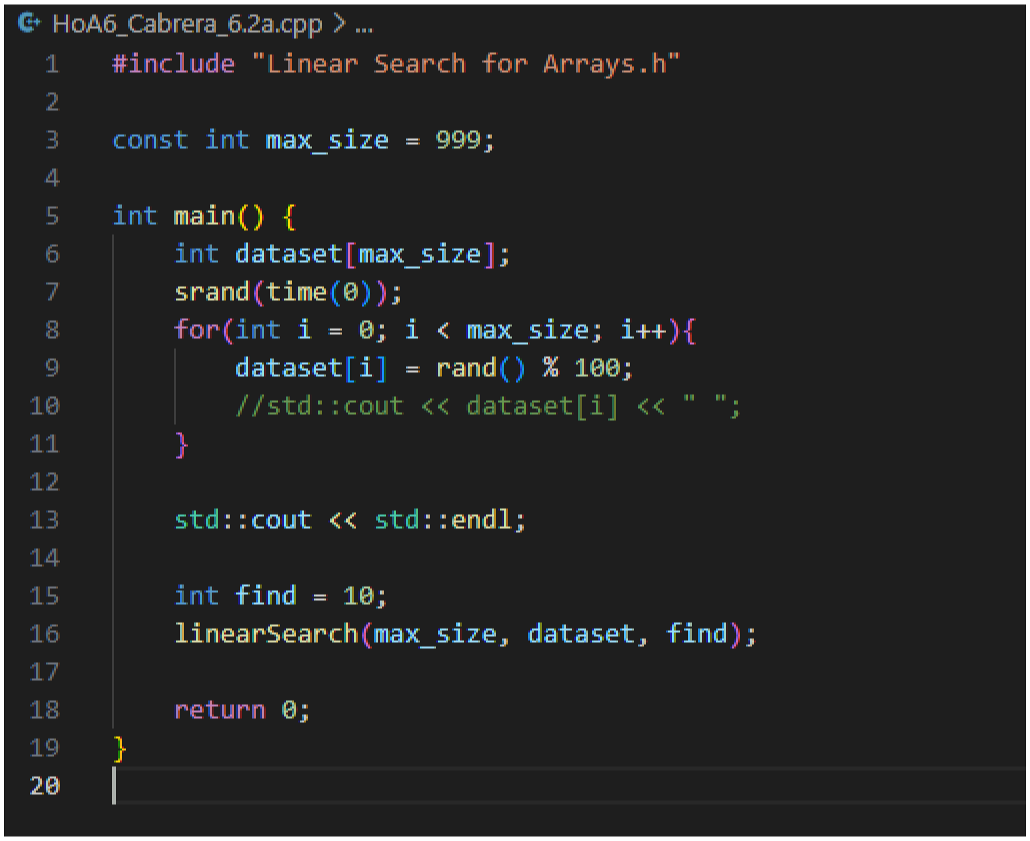
## 6. Output

| | |
|---|---|
| Screenshot |  |
| Observations | By using the stl library to generate random integers, the code generated random numbers using a for loop until it reached the max_size that I set which is 999 for this instance. |

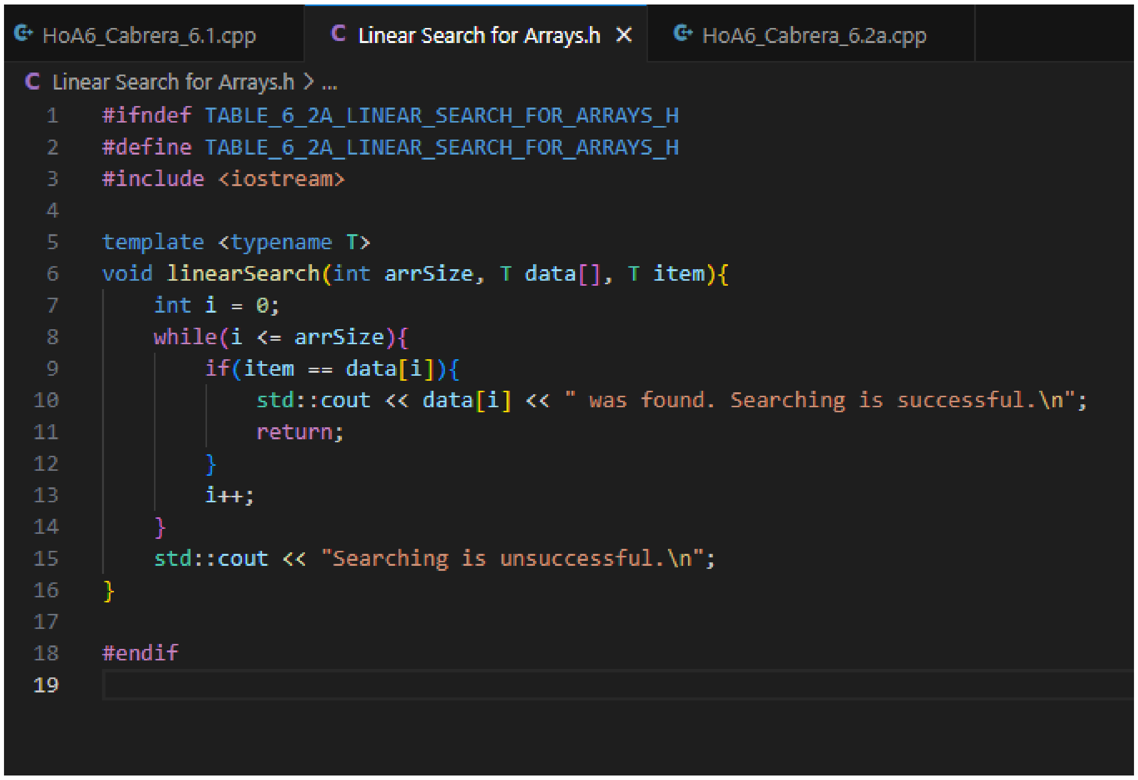
```
PS C:\Users\gabca>
PS C:\Users\gabca>  & 'c:\Users\gabca\.vscode\extensions\ms-vscode.cpptools-1.26.5-win32-x64\debugAdapters\bin\WindowsDebugLa
MIEngine-Out-3ylsrru2.32d' '--stderr=Microsoft-MIEngine-Error-t0lsusrf.5xj' '--pid=Microsoft-MIEngine-Pid-5tpgfxwh.pcr' '--db
1940 15876 24175 32104 32123 3388 31038 16857 26323 19827 25372 9972 26785 26855 4448 28806 3091 1635 2234 25936 15982 22883
10358 1886 927 31326 28754 27087 28024 19343 2503 19096 18150 16938 14979 8958 1323 6972 13550 24343 6674 7982 1631 25391 296
 1936 10982 12596 22795 20015 4603 24930 21878 31311 20663 26134 26086 18850 19481 30241 29811 4653 1312 17750 313 16622 1148
 13653 30451 2246 11191 25684 5663 28167 24037 10557 12621 27136 24498 10220 31036 21891 9595 18141 12229 19221 32639 30180 9
976 21125 21558 25538 3761 31453 16992 3824 10858 21810 6948 31899 27584 7011 11829 1469 12189 16197 19555 1269 16112 7065 12
5 30850 31405 15651 6672 22735 5293 20141 5649 25900 14259 12679 7424 2091 21654 16763 20621 32136 5560 9909 583 12902 20023
913 8555 25286 16030 16781 21191 30902 17155 5381 7346 6974 29390 30388 32432 20835 26694 18907 30928 23941 21653 21001 31537
7073 8886 27034 24402 32752 13813 13498 23717 9252 19475 25714 2178 15291 20227 11453 12684 23323 16552 2755 6106 23376 2537
208 6552 3063 19401 17971 27973 10577 17621 6397 10076 7913 18100 180 26749 12785 19607 27473 2064 14404 198 5777 22312 24497
6960 14840 27415 10721 1212 17777 2738 29501 25957 19845 21387 22455 21541 8437 7533 17345 28733 29775 21434 6658 25414 5654
 29095 17474 10121 4832 24071 12025 22303 14123 20800 18760 28556 28185 2323 29750 16070 15003 30270 12511 1933 16958 7851 18
6 14886 19714 1442 29268 27129 1416 14416 10443 12628 27773 21668 18302 17522 1726 10065 5729 328 22778 1697 9640 15886 3102
 25387 14311 14679 1872 29436 2529 26464 6281 22414 21679 22962 5286 2113 30910 23376 5547 1492 18271 15768 6033 19499 15739
5 19608 13481 21043 24721 2039 17399 15435 17795 11695 6750 4776 9459 6666 7826 28290 2891 24469 12667 9068 23263 35 4909 246
 23680 22372 12321 31772 15940 19313 5929 20407 12959 1310 31023 702 31610 15608 29082 13264 10893 13615 5101 23092 1162 2456
2815 18647 30835 13300 28644 18541 21172 3430 20784 26849 31887 25973 25181 6792 9294 29527 30622 13146 4192 19150 24692 2569
 14232 18662 15500 14359 6771 11030 28153 13080 13780 1288 26658 2021 23513 23284 28334 14997 21263 8387 3887 9858 15669 5575
27 25101 22403 28926 23338 19516 12276 20209 1728 1903 6590 16297 10645 16957 30894 2320 24055 10638 4758 13313 14953 9894 25
4 25172 14238 1331 15108 26895 12008 29218 7192 28023 19718 12039 21479 24009 2828 17720 10437 30 2951 11939 18486 14839 5051
8867 27643 15050 19078 28868 24546 1650 10934 23292 1059 26423 15638 11800 29542 18925 17588 17461 30048 5446 5138 21032 9896
```

Table 6-1. Data Generated and Observations.

Code

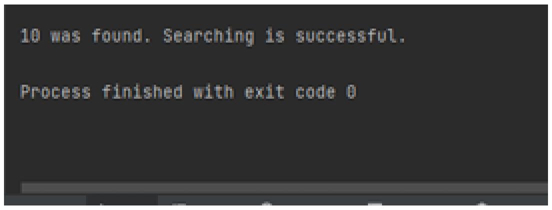


```cpp
#include "Linear Search for Arrays.h"

const int max_size = 999;

int main() {
    int dataset[max_size];
    srand(time(0));
    for(int i = 0; i < max_size; i++){
        dataset[i] = rand() % 100;
        //std::cout << dataset[i] << " ";
    }

    std::cout << std::endl;

    int find = 10;
    linearSearch(max_size, dataset, find);

    return 0;
}
```



```cpp
#ifndef TABLE_6_2A_LINEAR_SEARCH_FOR_ARRAYS_H
#define TABLE_6_2A_LINEAR_SEARCH_FOR_ARRAYS_H
#include <iostream>

template <typename T>
void linearSearch(int arrSize, T data[], T item){
    int i = 0;
    while(i <= arrSize){
        if(item == data[i]){
            std::cout << data[i] << " was found. Searching is successful.\n";
            return;
        }
        i++;
    }
    std::cout << "Searching is unsuccessful.\n";
}

#endif
```

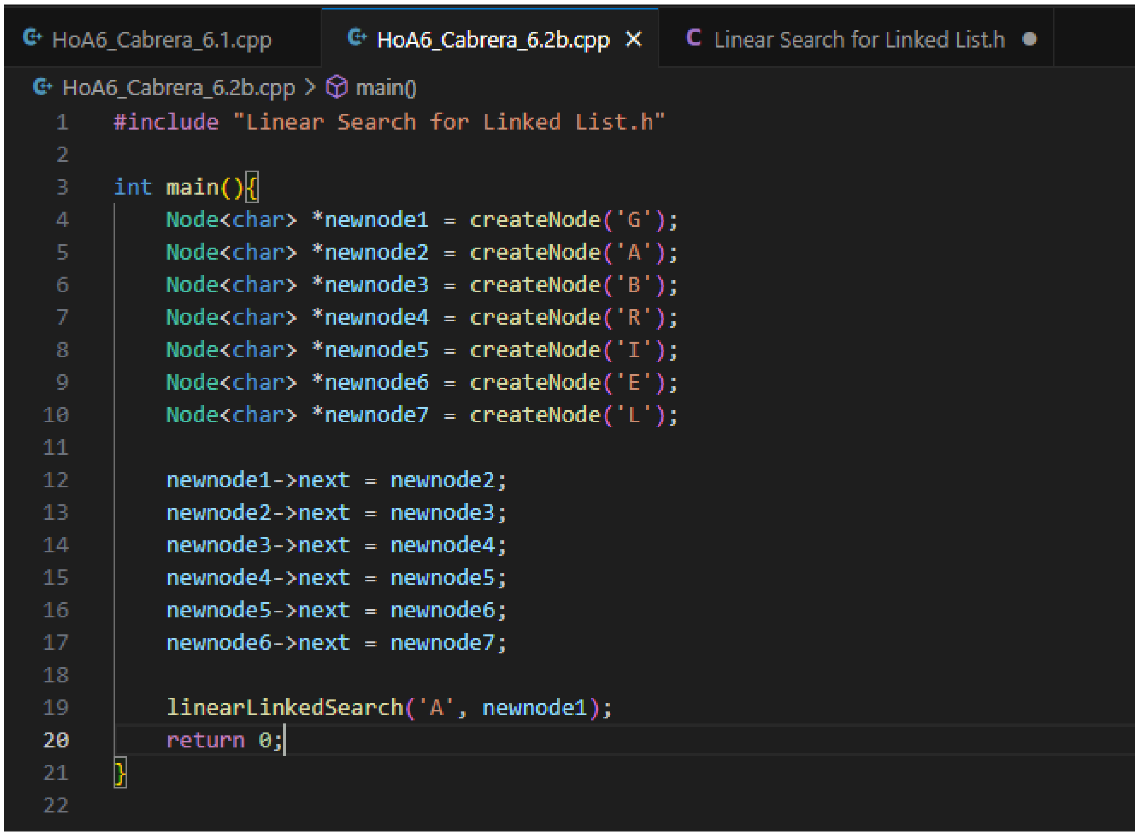
Output



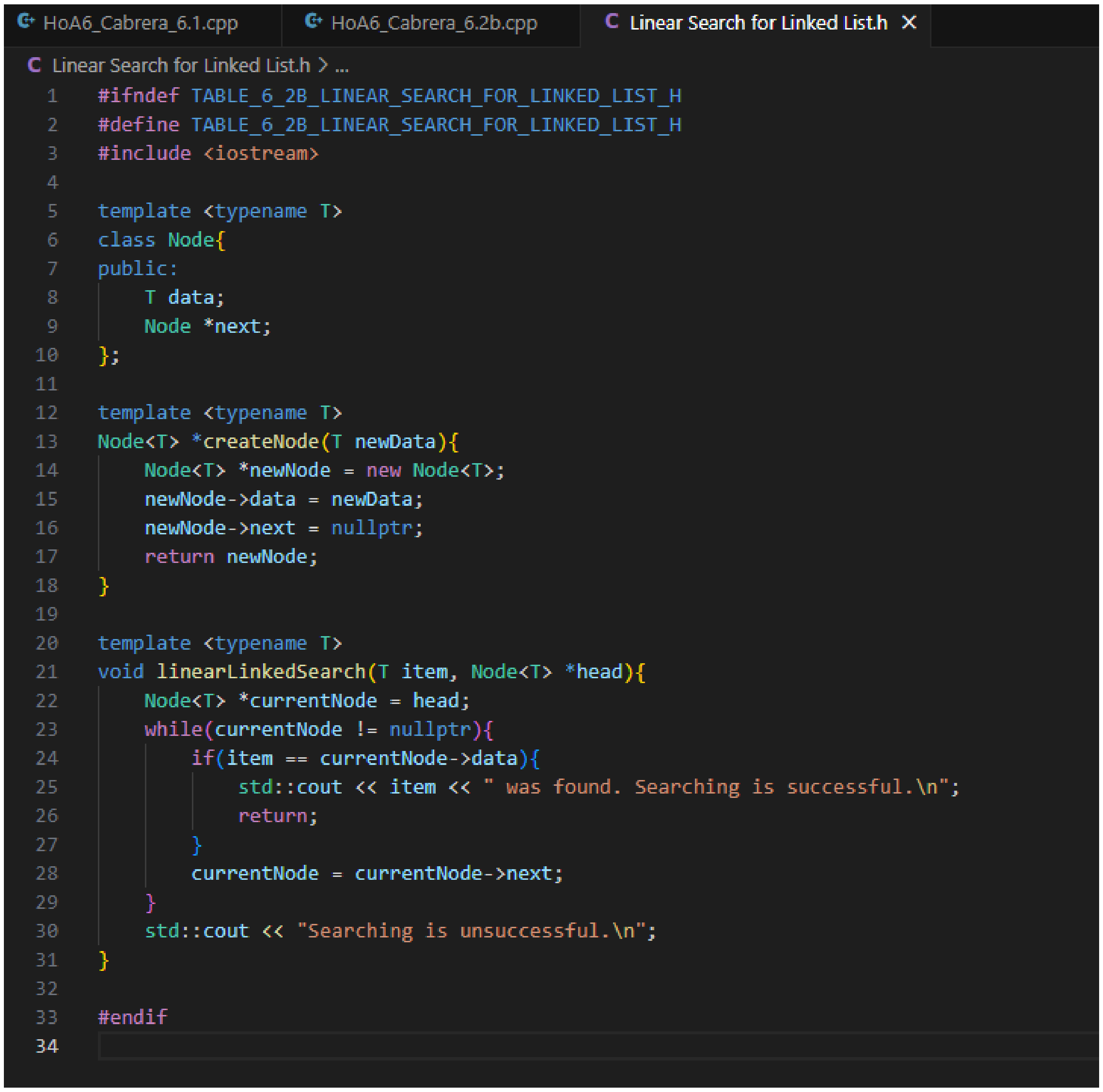
```
10 was found. Searching is successful.

Process finished with exit code 0
```

| | |
|---|---|
| Observations | Since the algorithm is linear search, as long as the data set is big enough, the program will have a high chance of generating the number 10, which the program will find eventually. |

Table 6-2a. Linear Search for Arrays

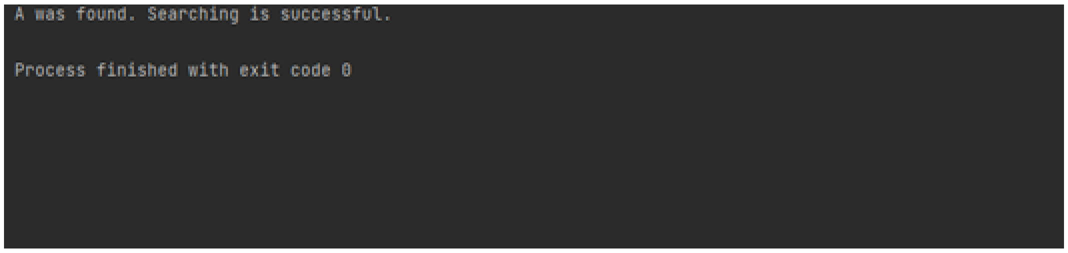
| | |
|---|---|
| Code |  HoA6_Cabrera_6.1.cpp · HoA6_Cabrera_6.2b.cpp · Linear Search for Linked List.h — HoA6_Cabrera_6.2b.cpp > main() |

```
#include "Linear Search for Linked List.h"

int main(){
    Node<char> *newnode1 = createNode('G');
    Node<char> *newnode2 = createNode('A');
    Node<char> *newnode3 = createNode('B');
    Node<char> *newnode4 = createNode('R');
    Node<char> *newnode5 = createNode('I');
    Node<char> *newnode6 = createNode('E');
    Node<char> *newnode7 = createNode('L');

    newnode1->next = newnode2;
    newnode2->next = newnode3;
    newnode3->next = newnode4;
    newnode4->next = newnode5;
    newnode5->next = newnode6;
    newnode6->next = newnode7;

    linearLinkedSearch('A', newnode1);
    return 0;
}

```

| | |
|---|---|
| |  |

HoA6_Cabrera_6.1.cpp
HoA6_Cabrera_6.2b.cpp
Linear Search for Linked List.h
Linear Search for Linked List.h > ...

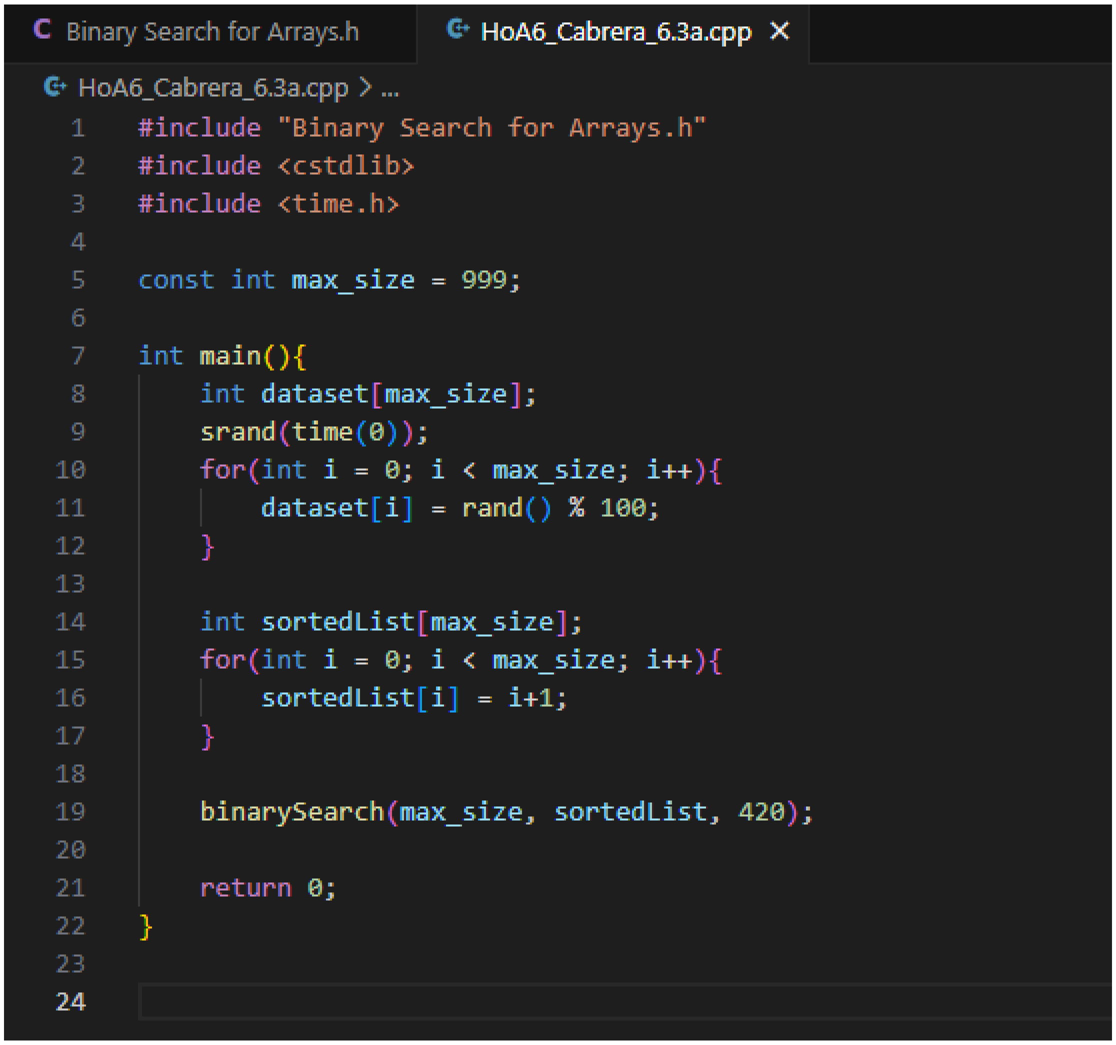
```cpp
#ifndef TABLE_6_2B_LINEAR_SEARCH_FOR_LINKED_LIST_H
#define TABLE_6_2B_LINEAR_SEARCH_FOR_LINKED_LIST_H
#include <iostream>

template <typename T>
class Node{
public:
    T data;
    Node *next;
};

template <typename T>
Node<T> *createNode(T newData){
    Node<T> *newNode = new Node<T>;
    newNode->data = newData;
    newNode->next = nullptr;
    return newNode;
}

template <typename T>
void linearLinkedSearch(T item, Node<T> *head){
    Node<T> *currentNode = head;
    while(currentNode != nullptr){
        if(item == currentNode->data){
            std::cout << item << " was found. Searching is successful.\n";
            return;
        }
        currentNode = currentNode->next;
    }
    std::cout << "Searching is unsuccessful.\n";
}

#endif
```

| | |
|---|---|
| Output |  |

```
A was found. Searching is successful.

Process finished with exit code 0
```

| | |
|---|---|
| Observations | For the linked list, I made a linked list of my own name and the program starts by checking each node for the letter I put in the search function, which is A. |

Table 6-2b. Linear Search for Linked List

| Code |  Binary Search for Arrays.h · HoA6_Cabrera_6.3a.cpp · HoA6_Cabrera_6.3a.cpp > ... |
|---|---|

```cpp
#include "Binary Search for Arrays.h"
#include <cstdlib>
#include <time.h>

const int max_size = 999;

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

    int sortedList[max_size];
    for(int i = 0; i < max_size; i++){
        sortedList[i] = i+1;
    }

    binarySearch(max_size, sortedList, 420);

    return 0;
}

```

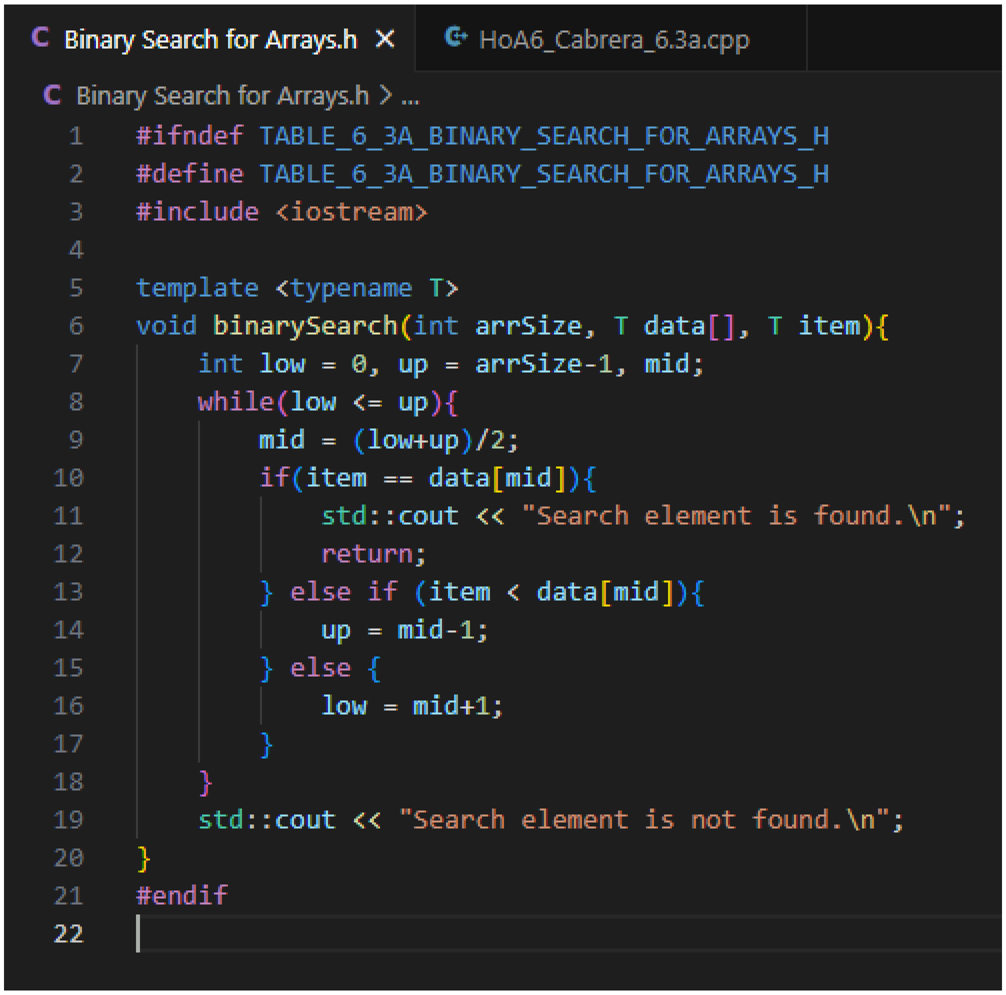
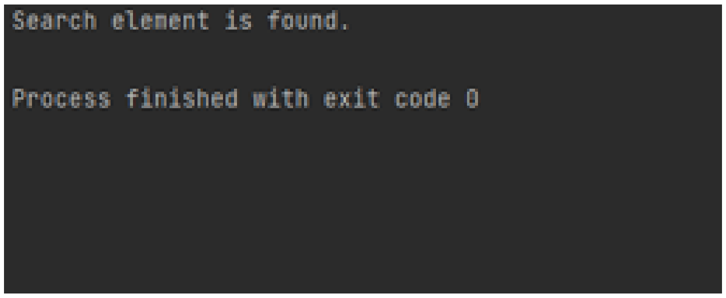
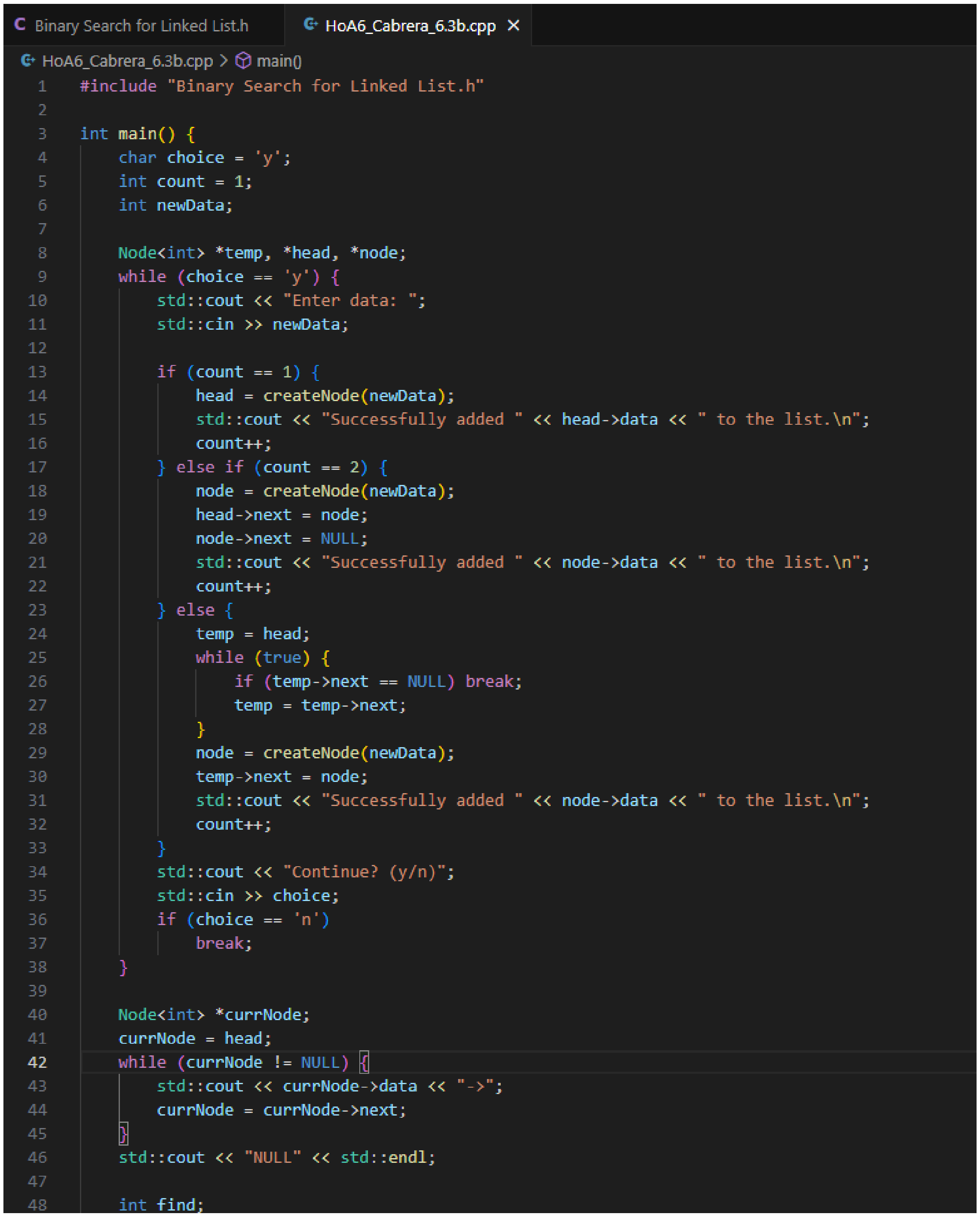
| | |
|---|---|
| |  Binary Search for Arrays.h ✕ HoA6_Cabrera_6.3a.cpp<br>Binary Search for Arrays.h > ...<br>`#ifndef TABLE_6_3A_BINARY_SEARCH_FOR_ARRAYS_H`<br>`#define TABLE_6_3A_BINARY_SEARCH_FOR_ARRAYS_H`<br>`#include <iostream>`<br><br>`template <typename T>`<br>`void binarySearch(int arrSize, T data[], T item){`<br>`    int low = 0, up = arrSize-1, mid;`<br>`    while(low <= up){`<br>`        mid = (low+up)/2;`<br>`        if(item == data[mid]){`<br>`            std::cout << "Search element is found.\n";`<br>`            return;`<br>`        } else if (item < data[mid]){`<br>`            up = mid-1;`<br>`        } else {`<br>`            low = mid+1;`<br>`        }`<br>`    }`<br>`    std::cout << "Search element is not found.\n";`<br>`}`<br>`#endif` |
| Output |  `Search element is found.`<br><br>`Process finished with exit code 0` |
| Observations | Similar to the linear search for arrays, assigning a large number to the max_size variable gives the program a high chance of generating and finding the search element. |

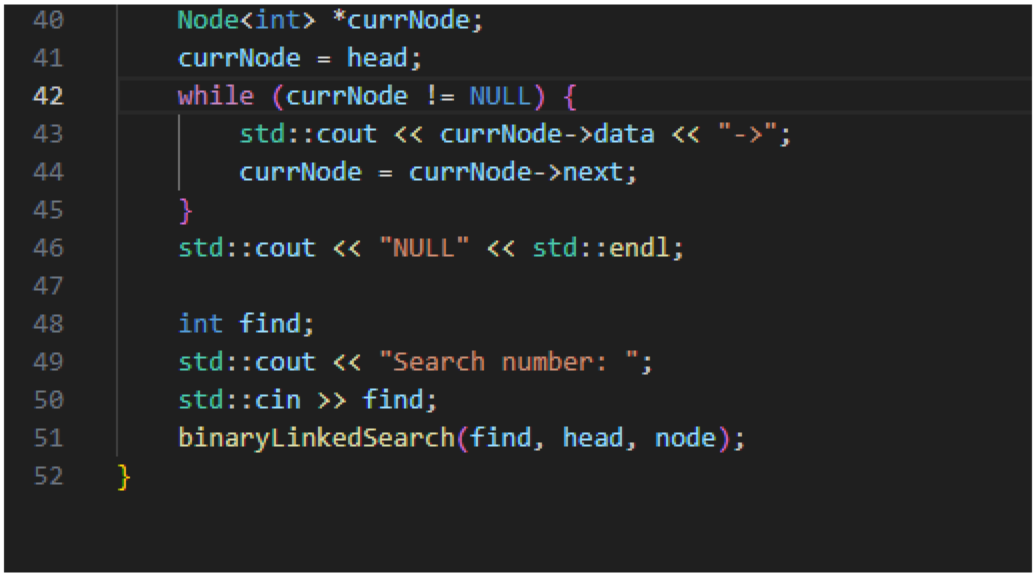
Table 6-3a. Binary Search for Arrays

| Code |  |
|---|---|

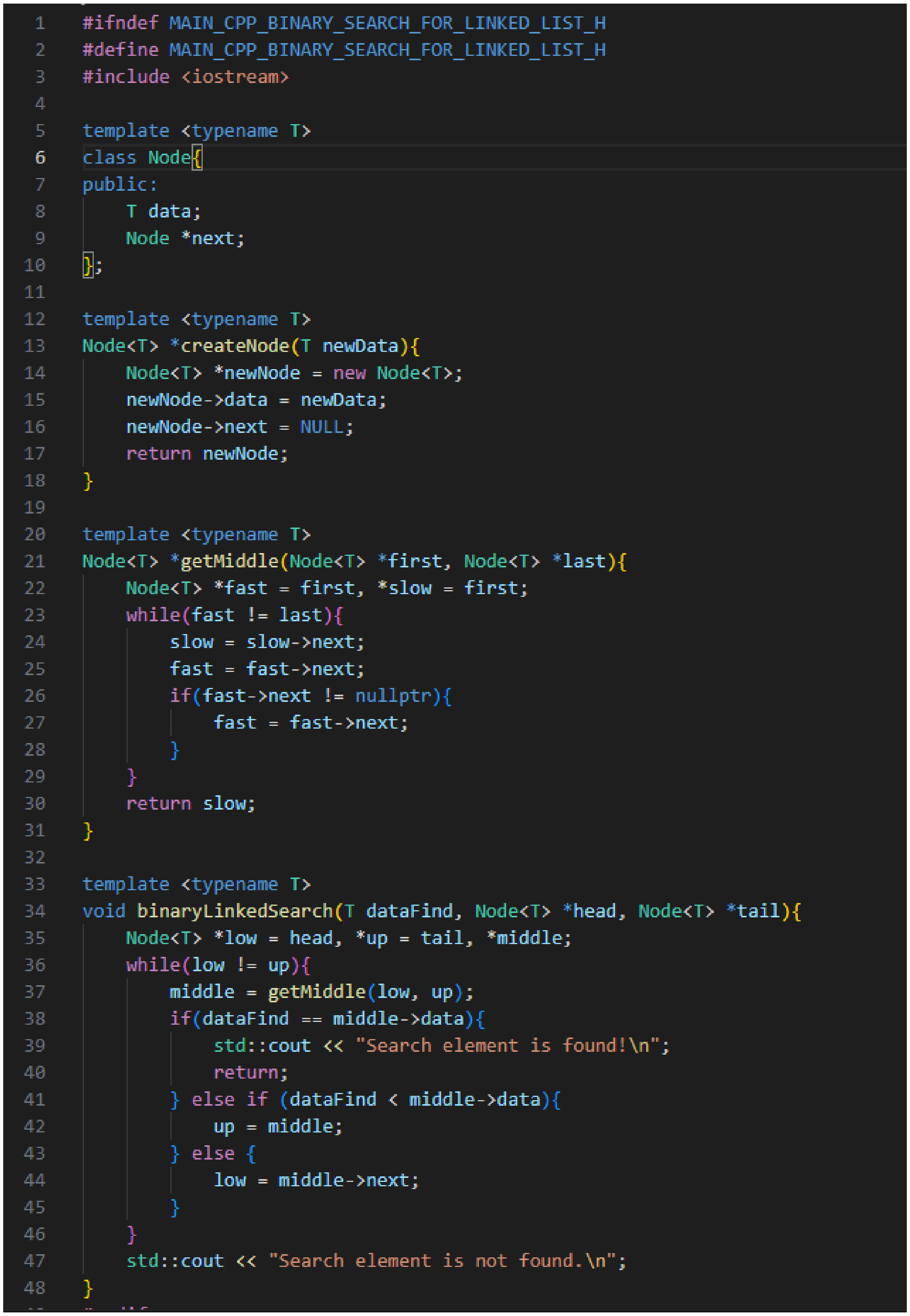
Binary Search for Linked List.h    HoA6_Cabrera_6.3b.cpp ×

HoA6_Cabrera_6.3b.cpp > main()

```
#include "Binary Search for Linked List.h"

int main() {
    char choice = 'y';
    int count = 1;
    int newData;

    Node<int> *temp, *head, *node;
    while (choice == 'y') {
        std::cout << "Enter data: ";
        std::cin >> newData;

        if (count == 1) {
            head = createNode(newData);
            std::cout << "Successfully added " << head->data << " to the list.\n";
            count++;
        } else if (count == 2) {
            node = createNode(newData);
            head->next = node;
            node->next = NULL;
            std::cout << "Successfully added " << node->data << " to the list.\n";
            count++;
        } else {
            temp = head;
            while (true) {
                if (temp->next == NULL) break;
                temp = temp->next;
            }
            node = createNode(newData);
            temp->next = node;
            std::cout << "Successfully added " << node->data << " to the list.\n";
            count++;
        }
        std::cout << "Continue? (y/n)";
        std::cin >> choice;
        if (choice == 'n')
            break;
    }

    Node<int> *currNode;
    currNode = head;
    while (currNode != NULL) {
        std::cout << currNode->data << "->";
        currNode = currNode->next;
    }
    std::cout << "NULL" << std::endl;

    int find;
```

| | |
|---|---|
| |  |

```cpp
    Node<int> *currNode;
    currNode = head;
    while (currNode != NULL) {
        std::cout << currNode->data << "->";
        currNode = currNode->next;
    }
    std::cout << "NULL" << std::endl;

    int find;
    std::cout << "Search number: ";
    std::cin >> find;
    binaryLinkedSearch(find, head, node);
}
```

| | |
|---|---|
| |  |

```cpp
#ifndef MAIN_CPP_BINARY_SEARCH_FOR_LINKED_LIST_H
#define MAIN_CPP_BINARY_SEARCH_FOR_LINKED_LIST_H
#include <iostream>

template <typename T>
class Node{
public:
    T data;
    Node *next;
};

template <typename T>
Node<T> *createNode(T newData){
    Node<T> *newNode = new Node<T>;
    newNode->data = newData;
    newNode->next = NULL;
    return newNode;
}

template <typename T>
Node<T> *getMiddle(Node<T> *first, Node<T> *last){
    Node<T> *fast = first, *slow = first;
    while(fast != last){
        slow = slow->next;
        fast = fast->next;
        if(fast->next != nullptr){
            fast = fast->next;
        }
    }
    return slow;
}

template <typename T>
void binaryLinkedSearch(T dataFind, Node<T> *head, Node<T> *tail){
    Node<T> *low = head, *up = tail, *middle;
    while(low != up){
        middle = getMiddle(low, up);
        if(dataFind == middle->data){
            std::cout << "Search element is found!\n";
            return;
        } else if (dataFind < middle->data){
            up = middle;
        } else {
            low = middle->next;
        }
    }
    std::cout << "Search element is not found.\n";
}
```

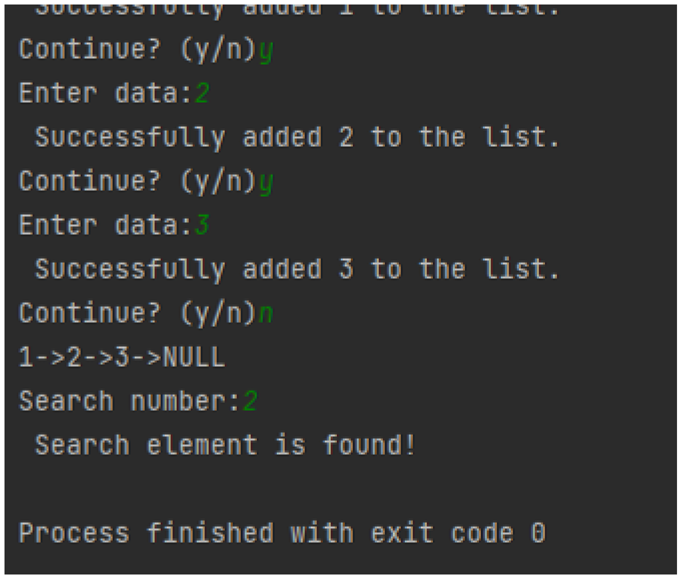
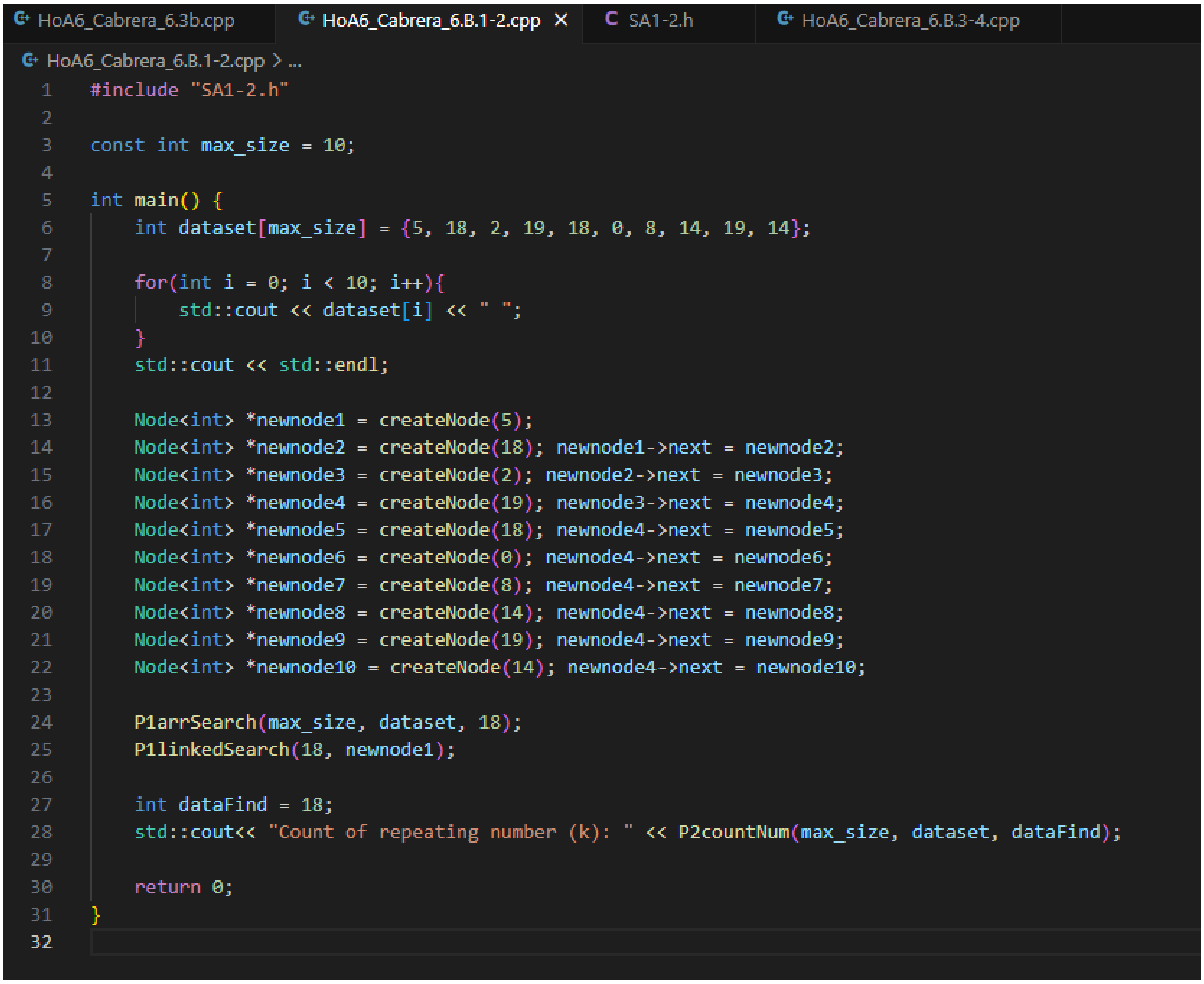
| | |
|---|---|
| Output | <br>` Successfully added 1 to the list.`<br>`Continue? (y/n)y`<br>`Enter data:2`<br>` Successfully added 2 to the list.`<br>`Continue? (y/n)y`<br>`Enter data:3`<br>` Successfully added 3 to the list.`<br>`Continue? (y/n)n`<br>`1->2->3->NULL`<br>`Search number:2`<br>` Search element is found!`<br><br>`Process finished with exit code 0` |
| Observations | The code checks first if the search element is the middle element, this time it is. If the middle element doesn’t match, it would check the left or right variables, depending if the middle element is higher or lower than the search element. |

Table 6-3b. Binary Search for Linked List

**7. Supplementary Activity**

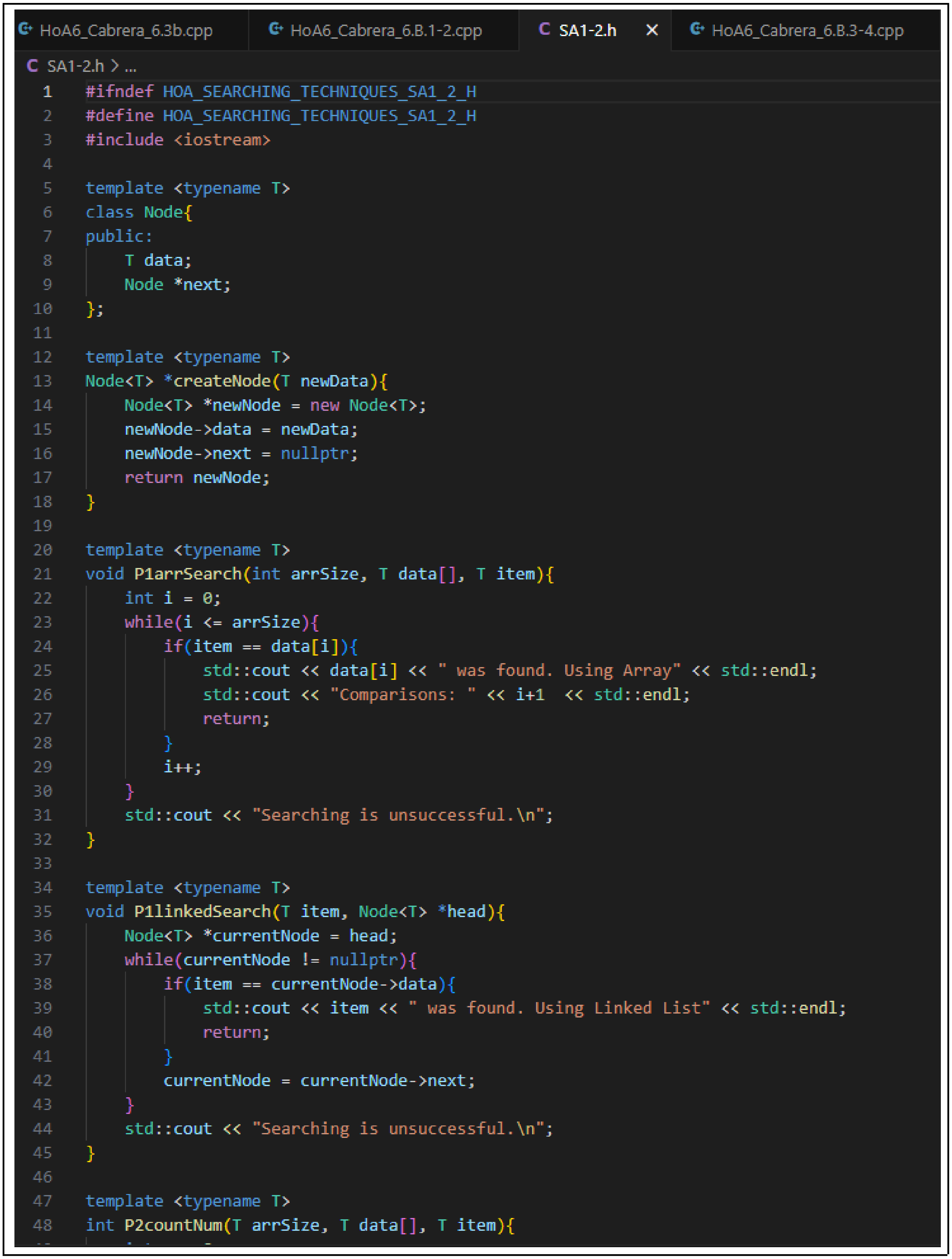
**Supplementary Activity 1 and 2**
**Code**



HoA6_Cabrera_6.3b.cpp | HoA6_Cabrera_6.B.1-2.cpp | SA1-2.h | HoA6_Cabrera_6.B.3-4.cpp

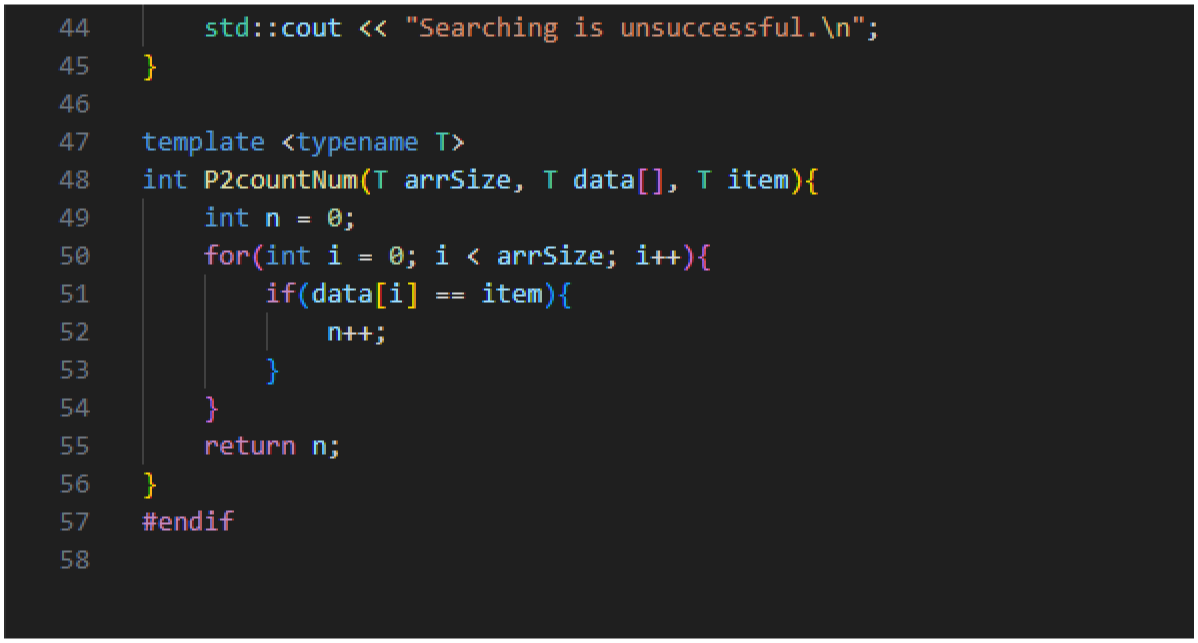
HoA6_Cabrera_6.B.1-2.cpp > ...

```cpp
#include "SA1-2.h"

const int max_size = 10;

int main() {
    int dataset[max_size] = {5, 18, 2, 19, 18, 0, 8, 14, 19, 14};

    for(int i = 0; i < 10; i++){
        std::cout << dataset[i] << " ";
    }
    std::cout << std::endl;

    Node<int> *newnode1 = createNode(5);
    Node<int> *newnode2 = createNode(18); newnode1->next = newnode2;
    Node<int> *newnode3 = createNode(2); newnode2->next = newnode3;
    Node<int> *newnode4 = createNode(19); newnode3->next = newnode4;
    Node<int> *newnode5 = createNode(18); newnode4->next = newnode5;
    Node<int> *newnode6 = createNode(0); newnode4->next = newnode6;
    Node<int> *newnode7 = createNode(8); newnode4->next = newnode7;
    Node<int> *newnode8 = createNode(14); newnode4->next = newnode8;
    Node<int> *newnode9 = createNode(19); newnode4->next = newnode9;
    Node<int> *newnode10 = createNode(14); newnode4->next = newnode10;

    P1arrSearch(max_size, dataset, 18);
    P1linkedSearch(18, newnode1);

    int dataFind = 18;
    std::cout<< "Count of repeating number (k): " << P2countNum(max_size, dataset, dataFind);

    return 0;
}

```

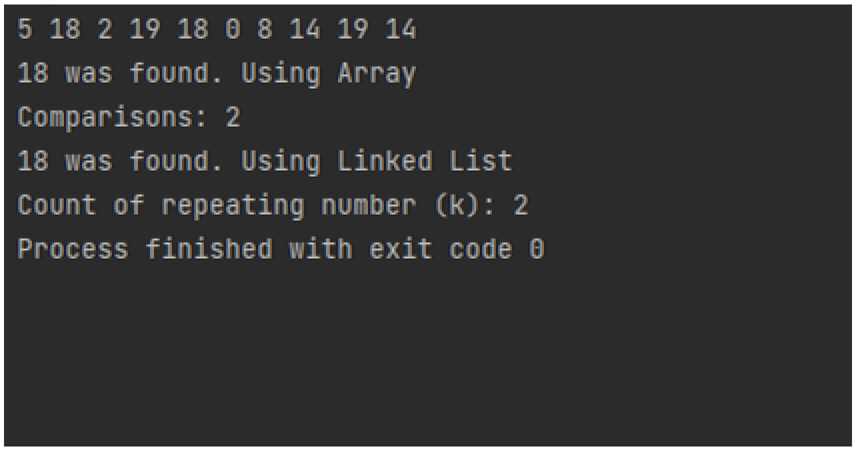

HoA6_Cabrera_6.3b.cpp | HoA6_Cabrera_6.B.1-2.cpp | SA1-2.h | HoA6_Cabrera_6.B.3-4.cpp

SA1-2.h > ...

```cpp
#ifndef HOA_SEARCHING_TECHNIQUES_SA1_2_H
#define HOA_SEARCHING_TECHNIQUES_SA1_2_H
#include <iostream>

template <typename T>
class Node{
public:
    T data;
    Node *next;
};

template <typename T>
Node<T> *createNode(T newData){
    Node<T> *newNode = new Node<T>;
    newNode->data = newData;
    newNode->next = nullptr;
    return newNode;
}

template <typename T>
void P1arrSearch(int arrSize, T data[], T item){
    int i = 0;
    while(i <= arrSize){
        if(item == data[i]){
            std::cout << data[i] << " was found. Using Array" << std::endl;
            std::cout << "Comparisons: " << i+1  << std::endl;
            return;
        }
        i++;
    }
    std::cout << "Searching is unsuccessful.\n";
}

template <typename T>
void P1linkedSearch(T item, Node<T> *head){
    Node<T> *currentNode = head;
    while(currentNode != nullptr){
        if(item == currentNode->data){
            std::cout << item << " was found. Using Linked List" << std::endl;
            return;
        }
        currentNode = currentNode->next;
    }
    std::cout << "Searching is unsuccessful.\n";
}

template <typename T>
int P2countNum(T arrSize, T data[], T item){
```


```
    std::cout << "Searching is unsuccessful.\n";
}

template <typename T>
int P2countNum(T arrSize, T data[], T item){
    int n = 0;
    for(int i = 0; i < arrSize; i++){
        if(data[i] == item){
            n++;
        }
    }
    return n;
}
#endif
```

**Output**


```
5 18 2 19 18 0 8 14 19 14
18 was found. Using Array
Comparisons: 2
18 was found. Using Linked List
Count of repeating number (k): 2
Process finished with exit code 0
```

**Supplementary Activity 3 and 4**

## No. 3-4 HOA 6.1 Algorithm Drawing

1. The array in which searching is to be performed:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 5 | 6 | 8 | 11 | 12 | 14 | 15 | 17 | 18 |

Let num = 8 be the element to be searched

2. Set two pointers low and high at the lowest and the highest positions respectively.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 5 | 6 | 8 | 11 | 12 | 14 | 15 | 17 | 18 |

LOW HIGH

3. Find the middle element "mid" of the array.
   "arr[low+(high-low)/2]=9"
4. If num==mid, then return mid. Else, compare the element to be searched with mid.
5. If num>mid, compare num with the middle element of the elements on the right side of mid. This is done by setting low to low = mid + 1.
6. Else, compare num with the middle element of the elements on the left side of mid. This is done by setting high to high = mid – 1.

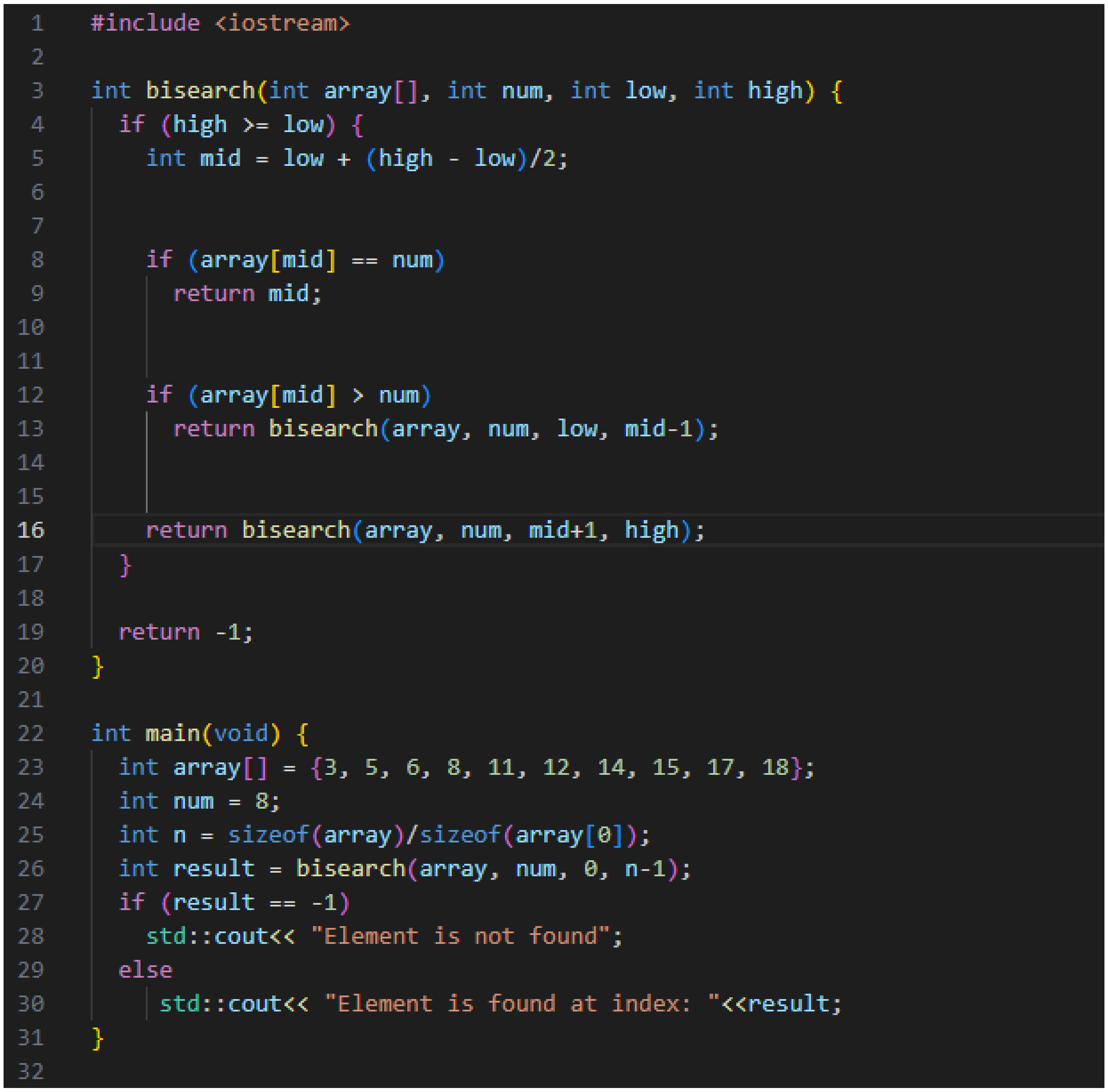
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 5 | 6 | 8 | 11 | 12 | 14 | 15 | 17 | 18 |
| | | | MID | | | | | | |

6. Else, compare num with the middle element of the elements on the left side of mid. This is done by setting high to high = mid – 1.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 5 | 6 | 8 | 11 | 12 | 14 | 15 | 17 | 18 |
| | | | MID | | | | | | |

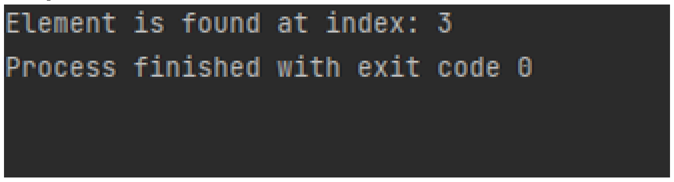
7. num = 8 is found

**Code**



```cpp
#include <iostream>

int bisearch(int array[], int num, int low, int high) {
  if (high >= low) {
    int mid = low + (high - low)/2;


    if (array[mid] == num)
      return mid;


    if (array[mid] > num)
      return bisearch(array, num, low, mid-1);


    return bisearch(array, num, mid+1, high);
  }

  return -1;
}

int main(void) {
  int array[] = {3, 5, 6, 8, 11, 12, 14, 15, 17, 18};
  int num = 8;
  int n = sizeof(array)/sizeof(array[0]);
  int result = bisearch(array, num, 0, n-1);
  if (result == -1)
    std::cout<< "Element is not found";
  else
     std::cout<< "Element is found at index: "<<result;
}

```

**Output**



```
Element is found at index: 3
Process finished with exit code 0
```

## 8. Conclusion

The program has successfully demonstrated how linear and binary search works and how to implement them using arrays and linked lists. I was able to make headers for the linear search and binary search for both arrays and linked lists which strengthened my knowledge more in those areas. I was also able to make an algorithm to make a function search for an element and make this function recursive. Overall this activity has given me a stronger foundation for programming that I can use in future projects.

## 9. Assessment Rubric