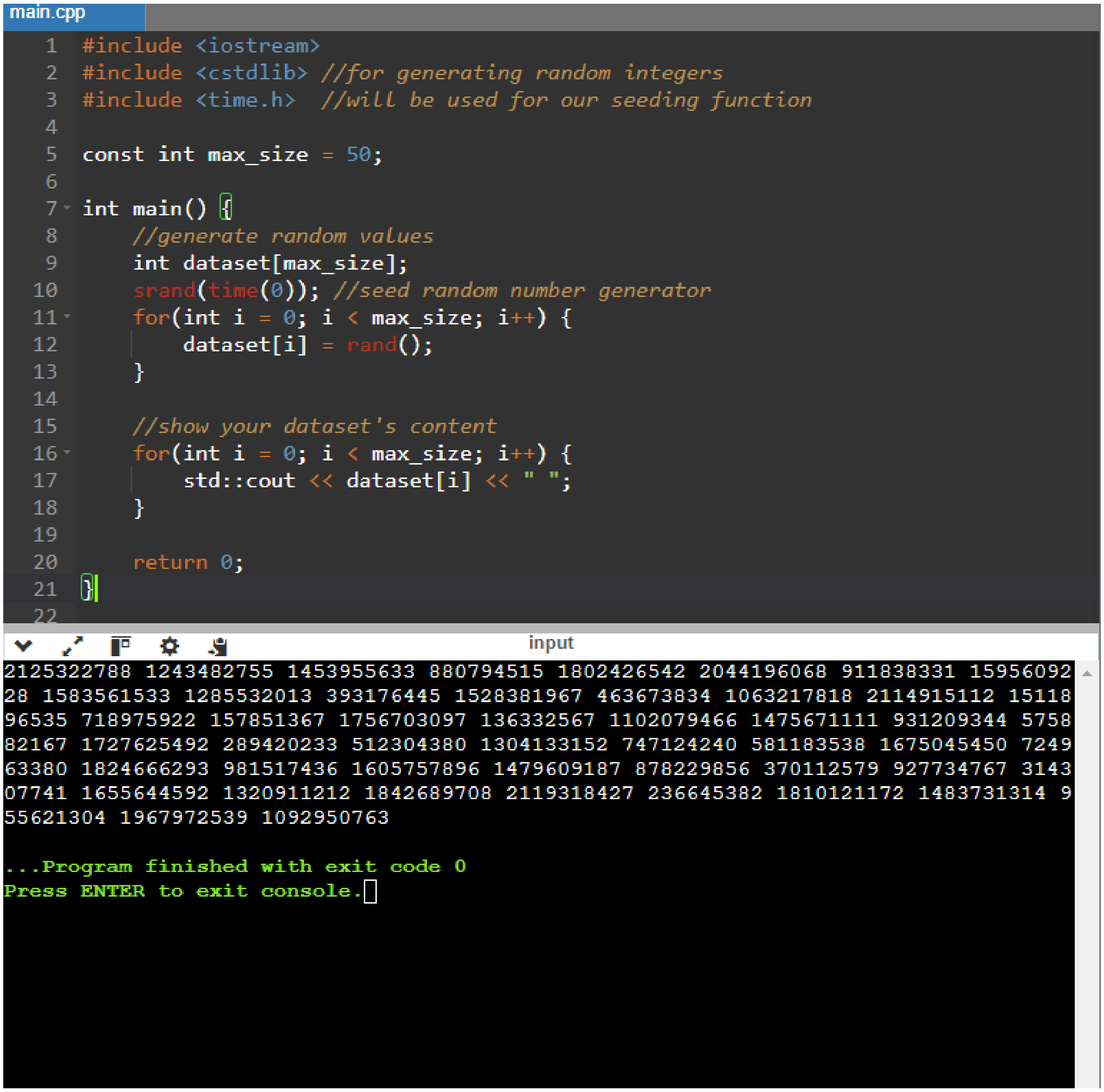
Activity No. <6>

SEARCHING TECHNIQUES

Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:10/15/2024**
Section: CPE21S4	**Date Submitted: 10/15/2024**
Name(s): Santos, Ma. Kassandra Nicole D.	**Instructor: Ma'am Rizette Sayo**

6. Output

Screenshot



```
#include <iostream>
#include <cstdlib> //for generating random integers
#include <time.h>  //will be used for our seeding function

const int max_size = 50;

int main() {
    //generate random values
    int dataset[max_size];
    srand(time(0)); //seed random number generator
    for(int i = 0; i < max_size; i++) {
        dataset[i] = rand();
    }

    //show your dataset's content
    for(int i = 0; i < max_size; i++) {
        std::cout << dataset[i] << " ";
    }

    return 0;
}
```

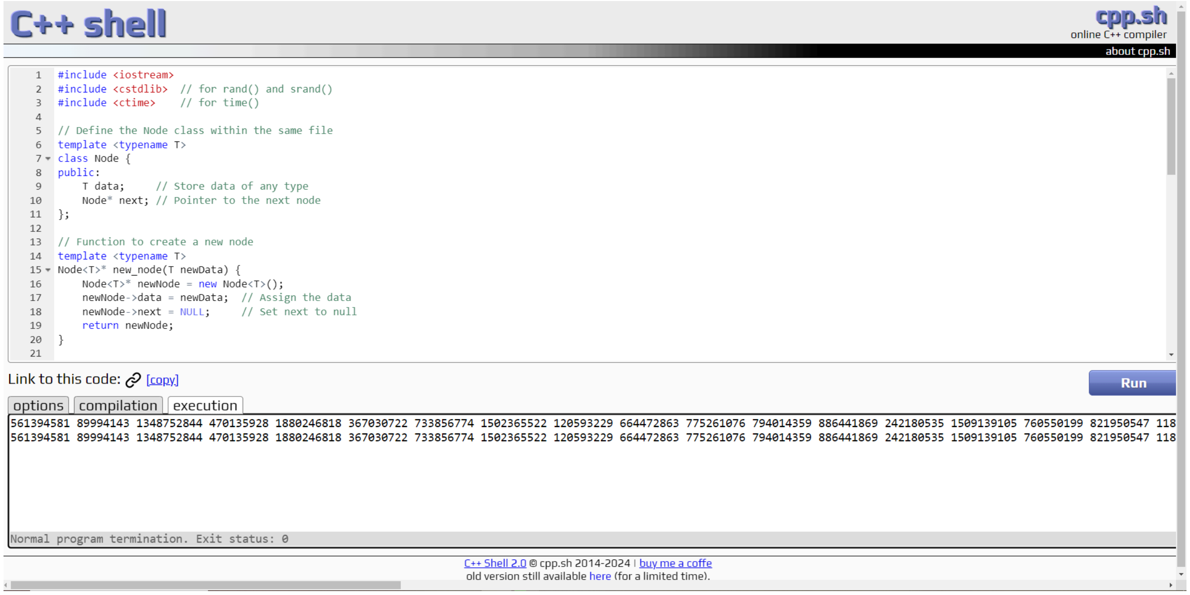
```
2125322788 1243482755 1453955633 880794515 1802426542 2044196068 911838331 15956092
28 1583561533 1285532013 393176445 1528381967 463673834 1063217818 2114915112 15118
96535 718975922 157851367 1756703097 136332567 1102079466 1475671111 931209344 5758
82167 1727625492 289420233 512304380 1304133152 747124240 581183538 1675045450 7249
63380 1824666293 981517436 1605757896 1479609187 878229856 370112579 927734767 3143
07741 1655644592 1320911212 1842689708 2119318427 236645382 1810121172 1483731314 9
55621304 1967972539 1092950763

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

Observation

I observed that once the code was excuted, it shows 50 random numbers

<table>
<tr>
<td>Screenshot and Output</td>
<td>



```cpp
#include <iostream>
#include <cstdlib>  // for rand() and srand()
#include <ctime>    // for time()

// Define the Node class within the same file
template <typename T>
class Node {
public:
    T data;     // Store data of any type
    Node* next; // Pointer to the next node
};

// Function to create a new node
template <typename T>
Node<T>* new_node(T newData) {
    Node<T>* newNode = new Node<T>();
    newNode->data = newData;  // Assign the data
    newNode->next = NULL;     // Set next to null
    return newNode;
}

```

Link to this code: [copy]

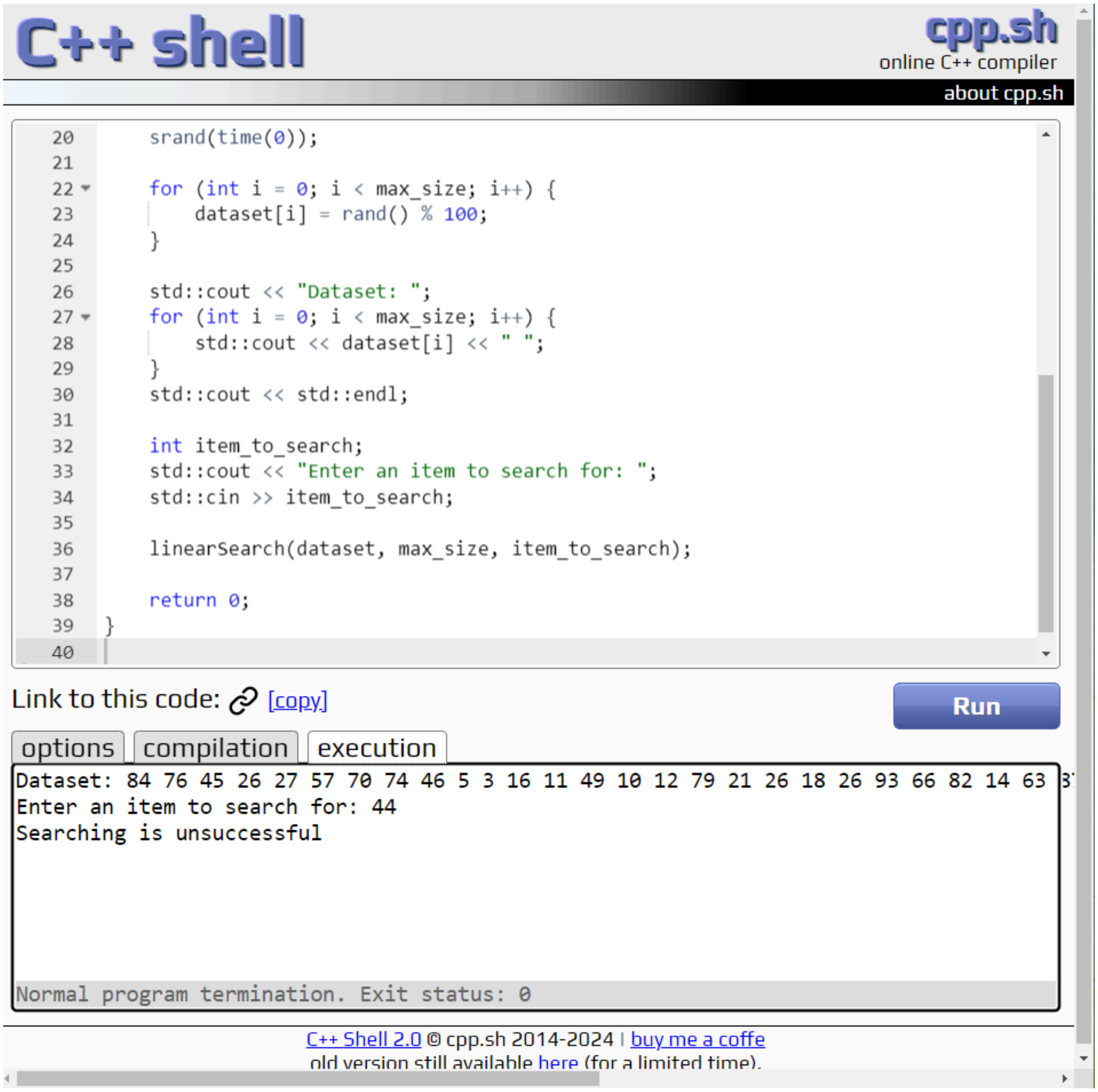
```
561394581 89994143 1348752844 470135928 1880246818 367030722 733856774 1502365522 120593229 664472863 775261076 794014359 886441869 242180535 1509139105 760550199 821950547 118
561394581 89994143 1348752844 470135928 1880246818 367030722 733856774 1502365522 120593229 664472863 775261076 794014359 886441869 242180535 1509139105 760550199 821950547 118
```

```
Normal program termination. Exit status: 0
```

C++ Shell 2.0 © cpp.sh 2014-2024 | buy me a coffe
old version still available here (for a limited time).

</td>
</tr>
<tr>
<td>Observation</td>
<td>From my observation, I noticed that the printed out a set of elements</td>
</tr>
</table>

Screenshot and Output


C++ shell
cpp.sh
online C++ compiler
about cpp.sh
Link to this code: [copy]
Run
options
compilation
execution
C++ Shell 2.0 © cpp.sh 2014-2024 | buy me a coffe
old version still available here (for a limited time).

```cpp
    srand(time(0));

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

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

    int item_to_search;
    std::cout << "Enter an item to search for: ";
    std::cin >> item_to_search;

    linearSearch(dataset, max_size, item_to_search);

    return 0;
}

```

```
Dataset: 84 76 45 26 27 57 70 74 46 5 3 16 11 49 10 12 79 21 26 18 26 93 66 82 14 63 3
Enter an item to search for: 44
Searching is unsuccessful

Normal program termination. Exit status: 0
```

Observation

I noticed that in this code, it gave a random set of numbers and gave an option to search for a number, when I types 44, it said that searching is unsuccessful, this would mean that the element 44 is not within the dataset that was given

7. Supplementary Activity

Problem 1:

CODE	CONSOLE

C/C++

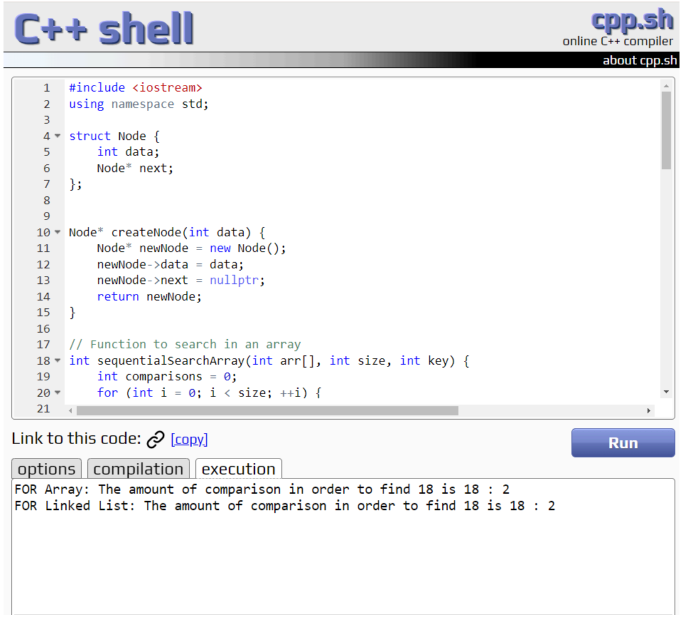
```cpp
#include <iostream>
using namespace std;

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


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

// Function to search in an array
int sequentialSearchArray(int arr[],
int size, int key) {
    int comparisons = 0;
    for (int i = 0; i < size; ++i) {
        comparisons++;
        if (arr[i] == key) {
            return comparisons;
        }
    }
    return comparisons;
}
// Function to search in a linked
list
int sequentialSearchLinkedList(Node*
head, int key) {
    int comparisons = 0;
    Node* current = head;
    while (current != nullptr) {
        comparisons++;
        if (current->data == key) {
            return comparisons;
        }
        current = current->next;
    }
    return comparisons;
}

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



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

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


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

// Function to search in an array
int sequentialSearchArray(int arr[], int size, int key) {
    int comparisons = 0;
    for (int i = 0; i < size; ++i) {

```

```
FOR Array: The amount of comparison in order to find 18 is 18 : 2
FOR Linked List: The amount of comparison in order to find 18 is 18 : 2
```

```
    int key = 18;

    // Create linked list
    Node* head = createNode(arr[0]);
    Node* current = head;
    for (int i = 1; i < size; ++i) {
        current->next =
createNode(arr[i]);
        current = current->next;
    }


    int arrayComparisons =
sequentialSearchArray(arr, size,
key);
    cout << "FOR Array: The amount
of comparison in order to find 18 is
" << key << " : " <<
arrayComparisons << endl;


    int listComparisons =
sequentialSearchLinkedList(head,
key);
    cout << "FOR Linked List: The
amount of comparison in order to
find 18 is " << key << " : " <<
listComparisons << endl;


    current = head;
    while (current != nullptr) {
        Node* temp = current;
        current = current->next;
        delete temp;
    }

    return 0;
}
```

Problem 2:

code	console

C/C++

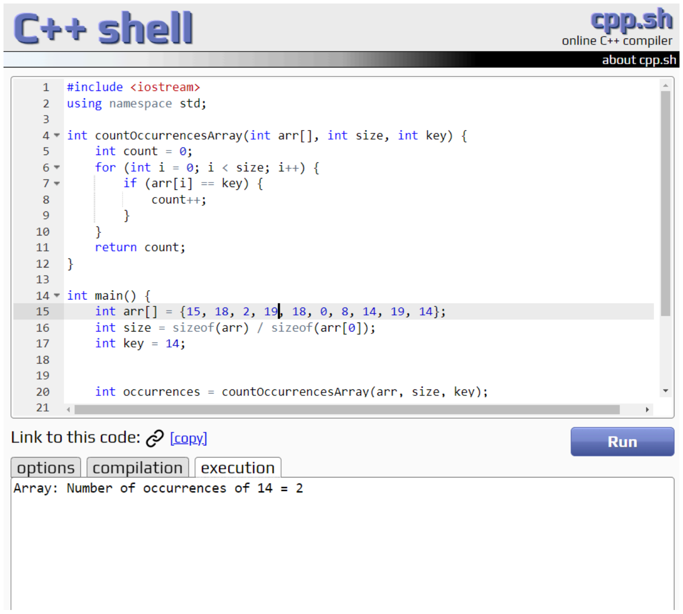
```cpp
#include <iostream>
using namespace std;

int countOccurrencesArray(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 = 14;


    int occurrences =
countOccurrencesArray(arr, size, key);
    cout << "Array: Number of
occurrences of " << key << " = " <<
occurrences << endl;

    return 0;
}
```



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

int countOccurrencesArray(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 = 14;


    int occurrences = countOccurrencesArray(arr, size, key);

```

```
Array: Number of occurrences of 14 = 2
```

Problem 3:

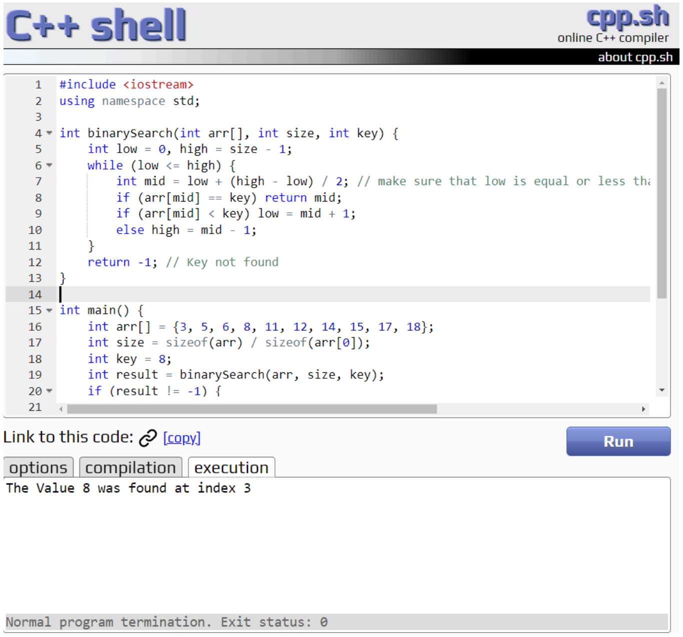
Code	console

C/C++

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

int binarySearch(int arr[], int size, int key) {
    int low = 0, high = size - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2; // make sure that low is equal or less than high, /2 is used  to find the middle index
        if (arr[mid] == key) return mid;
        if (arr[mid] < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1; // Key not found
}

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 = binarySearch(arr, size, key);
    if (result != -1) {
        cout << "The Value " << key << " was found at index " << result << endl;
    } else {
        cout << "THe Value " << key << " was not found " << endl;
    }
    return 0;
}
```



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

int binarySearch(int arr[], int size, int key) {
    int low = 0, high = size - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2; // make sure that low is equal or less tha
        if (arr[mid] == key) return mid;
        if (arr[mid] < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1; // Key not found
}

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 = binarySearch(arr, size, key);
    if (result != -1) {
```

```
The Value 8 was found at index 3
```

```
Normal program termination. Exit status: 0
```

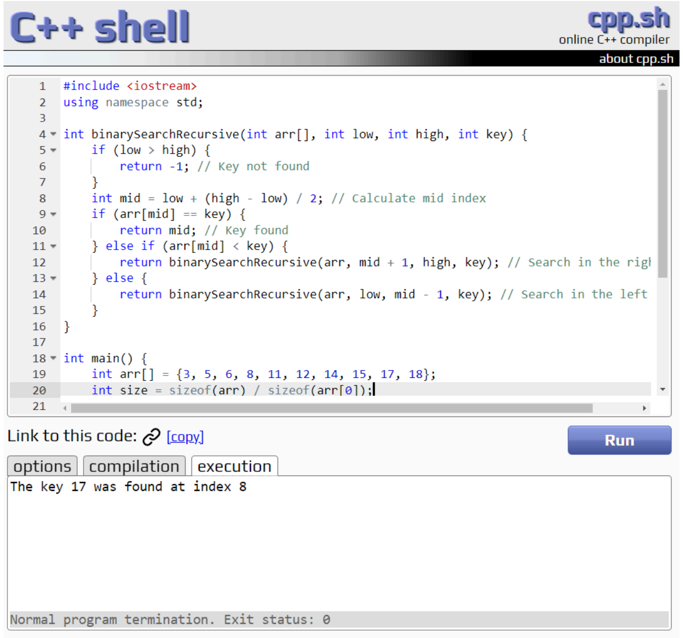
problem 4

code	console

```
C/C++
#include <iostream>
using namespace std;

int binarySearchRecursive(int arr[], int
low, int high, int key) {
    if (low > high) {
        return -1; // Key not found
    }
    int mid = low + (high - low) / 2; //
Calculate mid index
    if (arr[mid] == key) {
        return mid; // Key found
    } else if (arr[mid] < key) {
        return
binarySearchRecursive(arr, mid + 1,
high, key); // Search in the right half
    } else {
        return
binarySearchRecursive(arr, low, mid - 1,
key); // Search in the left half
    }
}

int main() {
    int arr[] = {3, 5, 6, 8, 11, 12, 14,
15, 17, 18};
    int size = sizeof(arr) /
sizeof(arr[0]);
    int key = 17;
    int result =
binarySearchRecursive(arr, 0, size - 1,
key);
    if (result != -1) {
        cout << "The key " << key << "
was found at index " << result << endl;
    } else {
        cout << "The key " << key << "
was not found" << endl;
    }
    return 0;
}
```



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

int binarySearchRecursive(int arr[], int low, int high, int key) {
    if (low > high) {
        return -1; // Key not found
    }
    int mid = low + (high - low) / 2; // Calculate mid index
    if (arr[mid] == key) {
        return mid; // Key found
    } else if (arr[mid] < key) {
        return binarySearchRecursive(arr, mid + 1, high, key); // Search in the rigl
    } else {
        return binarySearchRecursive(arr, low, mid - 1, key); // Search in the left
    }
}

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

```
The key 17 was found at index 8

Normal program termination. Exit status: 0
```

8. Conclusion

In this activity, we learned various ways to present numbers through linear search and binary search, binary search Searches a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array while the element in linear search must be found as the list is sequentially searched. For the supplementary activity, I noticed that most of the codes are similar in that it teach us various ways to search for a specific element within an array

or linked list. I think that overall, I did a good job in analyzing the code so far though I still think that I need to focus on learning how each function works as we move to a more complex lesson
9. Assessment Rubric