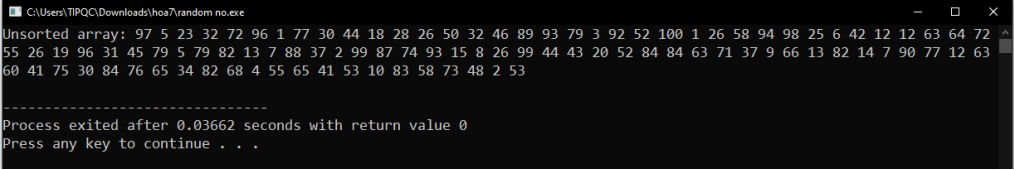


Activity No. 7	
SORTING ALGORITHMS: BUBBLE, SELECTION, AND INSERTION SORT	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 10/16/24
Section: CPE 21S4	Date Submitted:10/16/24
Name(s):Dominic Joseph P. Virtucio	Instructor: Ma'am Sayo
6. Output	
Code + Console Screenshot	<pre> C/C++ #include <iostream> #include <cstdlib> #include <ctime> using namespace std; int main() { srand(time(0)); int arr[100]; for (int i = 0; i < 100; i++) { arr[i] = rand() % 100 + 1; } cout << "Unsorted array: "; for (int i = 0; i < 100; i++) { cout << arr[i] << " "; } cout << endl; return 0; } </pre> 
Observations	The code produces an array of 100 random numbers ranging from 1 to 100, then outputs the unsorted array to the terminal. The result displays the generated array of random integers, which was created mostly by loops.
Table 7-1. Array of Values for Sort Algorithm Testing	

Code + Console Screenshot

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include "sorting_algorithms.h"

using namespace std;

int main() {
    srand(time(0));

    int arr[100];

    for (int i = 0; i < 100; i++) {
        arr[i] = rand() % 100 + 1;
    }

    cout << "Unsorted array: ";
    for (int i = 0; i < 100; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    bubbleSort(arr, 100);

    cout << "\n" << "Sorted array: ";
    for (int i = 0; i < 100; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

with implementation of bubble sorting

```
#ifndef SORTING_ALGORITHMS_H
#define SORTING_ALGORITHMS_H

#include <iostream>
#include <algorithm>

// Bubble Sort
template <typename T>
void bubbleSort(T arr[], size_t arrSize) {
    for (int i = 0; i < arrSize; i++) {
        for (int j = i + 1; j < arrSize; j++) {
            if (arr[j] < arr[i]) {
                std::swap(arr[j], arr[i]);
            }
        }
    }
}
```

output:

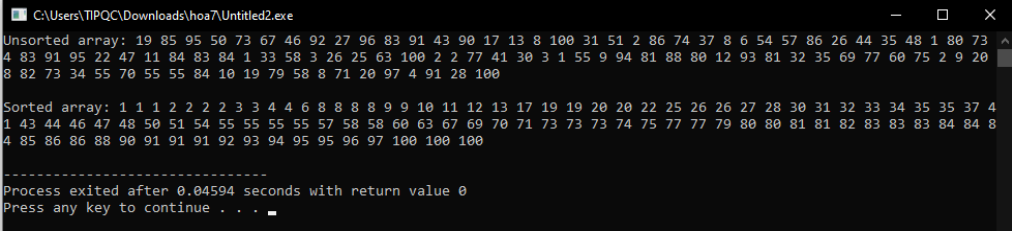
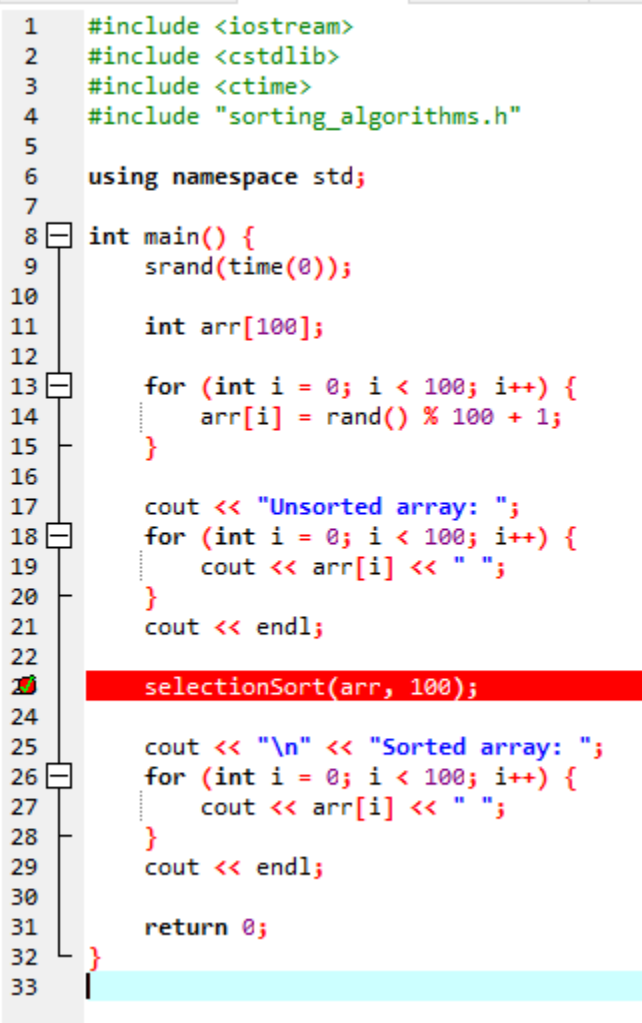
	
Observations	<p>As I go through the bubble sort algorithm, I notice how it painstakingly analyzes nearby components, switching them if they are out of order. Watching it in action, I observe the unsorted array gradually turn into a nicely sorted one, demonstrating the algorithm's ability to arrange elements in ascending order.</p>

Table 7-2. Bubble Sort Technique

Code + Console Screenshot	 <p>with implementation of selection sorting</p>
---------------------------	---

	<pre> // Selection Sort template <typename T> void selectionSort(T arr[], size_t arrSize) { for (int i = 0; i < arrSize - 1; i++) { int minIndex = i; for (int j = i + 1; j < arrSize; j++) { if (arr[j] < arr[minIndex]) { minIndex = j; } } std::swap(arr[minIndex], arr[i]); } } </pre>
Observations	<p>As I progress through the selection sort algorithm, this one repeatedly seeks for the least element in the unsorted area of the array and swaps it with the current index. Observing the result, I can easily observe the transition from unsorted to sorted array, demonstrating how rapidly the items are organized.</p>

Table 7-3. Selection Sort Algorithm

Code + Console Screenshot	<pre> 1 #include <iostream> 2 #include <cstdlib> 3 #include <ctime> 4 #include "sorting_algorithms.h" 5 6 using namespace std; 7 8 int main() { 9 srand(time(0)); 10 11 int arr[100]; 12 13 for (int i = 0; i < 100; i++) { 14 arr[i] = rand() % 100 + 1; 15 } 16 17 cout << "Unsorted array: "; 18 for (int i = 0; i < 100; i++) { 19 cout << arr[i] << " "; 20 } 21 cout << endl; 22 insertionSort(arr, 100); 23 24 cout << "\n" << "Sorted array: "; 25 for (int i = 0; i < 100; i++) { 26 cout << arr[i] << " "; 27 } 28 cout << endl; 29 30 return 0; 31 } 32 33 </pre> <p>with implementation of insertion sorting</p>
---------------------------	--

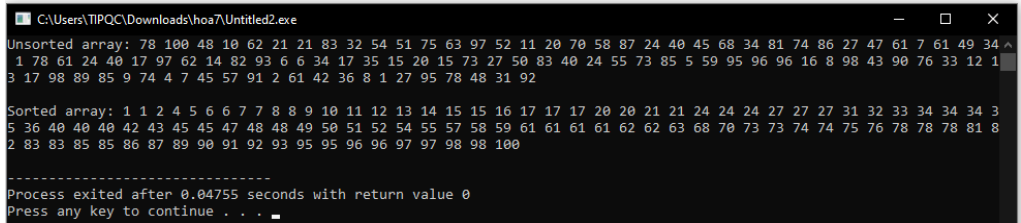
	<pre> // Insertion Sort template <typename T> void insertionSort(T arr[], size_t arrSize) { for (int i = 1; i < arrSize; i++) { int key = arr[i]; int j = i - 1; while (j >= 0 && arr[j] > key) { arr[j + 1] = arr[j]; j--; } arr[j + 1] = key; } } </pre> <p>Output:</p> 
Observations	<p>As I progress through the insertion sort method, the code implements the insertion sort algorithm, which successively constructs a sorted subarray by inserting each element from the unsorted section into its proper location within the sorted subarray. The result shows both the unsorted and sorted arrays, demonstrating the algorithm's effectiveness at sorting elements in ascending order.</p>

Table 7-4. Insertion Sort Algorithm

7. Supplementary Activity

```

C/C++
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <algorithm>

using namespace std;

int main() {
    srand(time(0));

    int votes[100];

    for (int i = 0; i < 100; i++) {
        votes[i] = rand() % 5 + 1;
    }

    cout << "Unsorted votes: ";
    for (int i = 0; i < 100; i++) {
        cout << votes[i] << " ";
    }
}

```

```

cout << endl;

sort(votes, votes + 100);

int candidate_counts[5] = {0};
for (int i = 0; i < 100; i++) {
    candidate_counts[votes[i] - 1]++;
}

int winning_candidate = 0;
int max_votes = candidate_counts[0];
for (int i = 1; i < 5; i++) {
    if (candidate_counts[i] > max_votes) {
        winning_candidate = i;
        max_votes = candidate_counts[i];
    }
}

string candidate_names[5] = {"Bo Dalton Capistrano", "Cornelius Raymon Agustin", "Deja Jayla Bañaga", "Lalla Brielle Yabut", "Franklin Relano Castro"};
string winning_candidate_name = candidate_names[winning_candidate];

cout << "Sorted votes: ";
for (int i = 0; i < 100; i++) {
    cout << votes[i] << " ";
}
cout << endl;

cout << "Vote counts for each candidate:" << endl;
for (int i = 0; i < 5; i++) {
    cout << "Candidate " << i + 1 << ": " << candidate_names[i] << " - " <<
candidate_counts[i] << " votes" << endl;
}

cout << "Winning candidate: " << winning_candidate_name << endl;

return 0;
}

```

Output console showing sorted array	Manual Count	Count Result of algorithm
Bubble sort Sorted votes: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 5	Vote counts for each candidate: Candidate 1: Bo Dalton Capistrano - 18 votes Candidate 2: Cornelius Raymon Agustin - 15 votes Candidate 3: Deja Jayla Bañaga - 14 votes Candidate 4: Lalla Brielle Yabut - 28 votes Candidate 5: Franklin Relano Castro - 25 votes	Winning candidate: Lalla Brielle Yabut
Selection sort Sorted votes: 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	Vote counts for each candidate: Candidate 1: Bo Dalton Capistrano - 20 votes Candidate 2: Cornelius Raymon Agustin - 19 votes Candidate 3: Deja Jayla Bañaga - 22 votes Candidate 4: Lalla Brielle Yabut - 19 votes Candidate 5: Franklin Relano Castro - 20 votes	Winning candidate: Deja Jayla Bañaga

Insertion sort

```
Sorted votes: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5
```

Vote counts for each candidate:

```
Candidate 1: Bo Dalton Capistrano - 25 votes
Candidate 2: Cornelius Raymon Agustin - 15
votes
Candidate 3: Deja Jayla Bañaga - 12 votes
Candidate 4: Lalla Brielle Yabut - 22 votes
Candidate 5: Franklin Relano Castro - 26
votes
```

Winning candidate: Franklin Relano Castro

Was your developed vote counting algorithm effective? Why or why not?

Answer: The developed algorithm for counting votes is indeed effective. By sorting the votes array, it precisely counts the votes for each candidate, showing how sorting algorithms can be used to make data processing tasks simpler.

8. Conclusion

I was able to effectively investigate the use and application of bubble sort, selection sort, and insertion sort—three basic sorting algorithms—during the lab exercise. I got the opportunity to examine each algorithm's performance using real-world cases, and they were all written in C++. All in all, this lab experience gave me insightful knowledge about the efficacy and efficiency of sorting algorithms in a range of computing applications. I now have a better grasp of these algorithms' operation as well as their useful applicability in real-world situations. It's amazing to observe how fundamental ideas like sorting are essential to more complex systems.