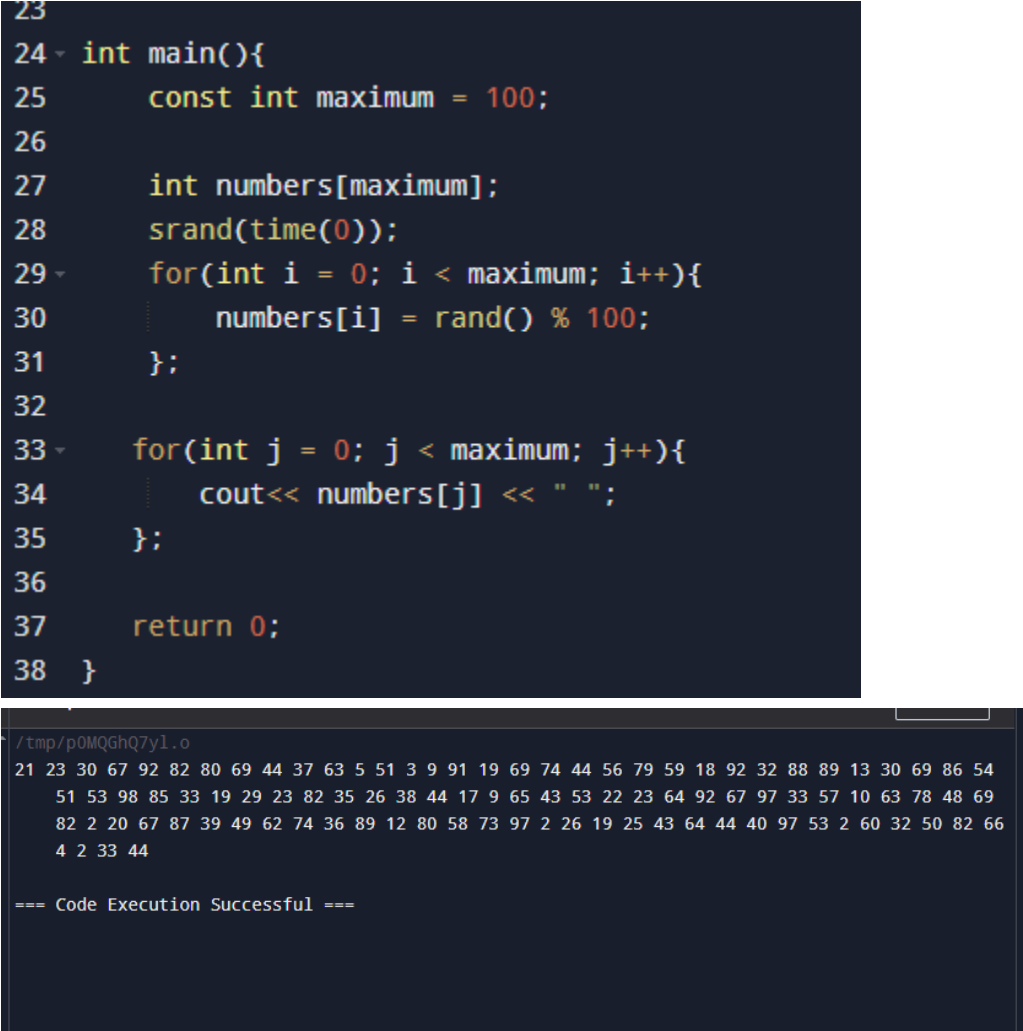


Activity No. 7	
Hands-on Activity 7.1 Sorting Algorithms	
Course Code: CPE021	Program: Computer Engineering
Course Title: Computer Architecture and Organization	Date Performed: 10/16/2024
Section: CPE21S1	Date Submitted: 10/17/2024
Name(s): Jhon Hendricks T. Bautista	Instructor: Mrs. Maria Rizette Sayo
A. Procedure: Output(s) and Observation(s)	
Code + Console Screenshot	 <pre> 23 24 int main(){ 25 const int maximum = 100; 26 27 int numbers[maximum]; 28 srand(time(0)); 29 for(int i = 0; i < maximum; i++){ 30 numbers[i] = rand() % 100; 31 }; 32 33 for(int j = 0; j < maximum; j++){ 34 cout<< numbers[j] << " "; 35 }; 36 37 return 0; 38 } </pre> <pre> /tmp/p0MQGhQ7y1.o 21 23 30 67 92 82 80 69 44 37 63 5 51 3 9 91 19 69 74 44 56 79 59 18 92 32 88 89 13 30 69 86 54 51 53 98 85 33 19 29 23 82 35 26 38 44 17 9 65 43 53 22 23 64 92 67 97 33 57 10 63 78 48 69 82 2 20 67 87 39 49 62 74 36 89 12 80 58 73 97 2 26 19 25 43 64 44 40 97 53 2 60 32 50 82 66 4 2 33 44 === Code Execution Successful === </pre>
Observation	<p>I observed in this preparation of code for sorting it outputted an unsorted array through the use of the method rand. This is commonly known for generating random numbers in C++ particularly in the cstdlib. The generated numbers are limited to 0 to 99 since and the limit of the array is 100 items.</p>
Table 7-1. Array of Values for Sort Algorithm Testing	

Code + Console Screenshot

```
main.cpp  [Icons] [Run]

8- void bubbleSort(T arr[], int arr_size){
9-     for(int x = 0; x < arr_size; x++){
10-         for(int y = 0; y < arr_size; y++){
11-             if(arr[y] > arr[y + 1]){
12-                 int temp = arr[y];
13-                 arr[y] = arr[y + 1];
14-                 arr[y + 1] = temp;
15-             };
16-         };
17-     };
18- };
19- };
20-
21-
22- void display(int arr[], int arr_size){
23-     for(int t = 0; t < arr_size; t++){
24-         cout<< arr[t] << " ";
25-     }
26- };
27-
28-
29-
30- int main(){
31-     const int maximum = 100;
32-
33-     int numbers[maximum];
34-     srand(time(0));
35-     for(int i = 0; i < maximum; i++){
36-         numbers[i] = rand() % 100;
37-     };
38-
39-     cout<<"UNSORTED ARRAY"<<endl;
40-     display(numbers, maximum);
41-
42-
43-     cout<<"\n";
44-
45-     cout<<"SORTED ARRAY"<<endl;
46-     bubbleSort(numbers, maximum);
47-     display(numbers, maximum);
48-
49- }
```

```
/tmp/sxFjFMPxeb.o
UNSORTED ARRAY
68 41 64 98 83 10 34 0 14 24 32 59 9 13 69 80 12 5 41 3 58 10 67 36 67 60 2 73 39 33 83 7 26 99 6 62 9 92 62
75 16 94 34 77 59 55 9 71 60 50 27 70 12 94 58 32 7 60 5 46 46 40 54 24 39 12 86 0 56 0 76 24 94 62 2 5
70 11 77 30 14 56 53 26 2 11 58 9 24 16 56 22 56 62 46 96 26 85 96 82
SORTED ARRAY
0 0 0 2 2 2 3 5 5 5 6 7 7 9 9 9 10 10 11 11 12 12 12 13 14 14 16 16 22 24 24 24 24 26 26 26 27 30 32 32 33
34 34 36 39 39 40 41 41 46 46 50 53 54 55 56 56 56 56 58 58 58 59 59 60 60 60 62 62 62 62 64 67 67 68
69 70 70 71 73 75 76 77 77 80 82 83 83 85 86 92 94 94 94 96 96 98 99

=== Code Execution Successful ===
```

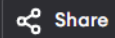
Observation

In this code for the bubble sort I observed that it uses a double for loop and an if condition for sorting the numbers in the array. The function uses the first element of the array and compares it to the next number. This continues to happen until it finds the number it is greater than. With that, all the numbers are traversed and successfully sorts the array of numbers.

Table 7-2. Bubble Sort Technique

Code + Console Screenshot

main.cpp



Run

```
1  #include <iostream>
2  #include <cstdlib>
3  #include <ctime>
4
5  using namespace std;
6
7  int routineSmallest(int arr[], int k, int arr_size){
8      int position, j;
9      int smallestElem = arr[k];
10     position = k;
11
12     for(int j = k+1; j < arr_size; j++){
13         if(arr[j] < smallestElem){
14             smallestElem = arr[j];
15             position = j;
16         }
17     }
18 }
19 return position;
20 };
21
22 void selectionSort(int arr[], int arr_size){
23     int pos, temp, pass = 0;
24     for(int i = 0; i < arr_size; i++){
25         pos = routineSmallest(arr, i, arr_size);
26         temp = arr[i];
27         arr[i] = arr[pos];
28         arr[pos] = temp;
29         pass++;
30     }
31 };
32
```

```

33
34
35 void display(int arr[], int arr_size){
36     for(int t = 0; t < arr_size; t++){
37         cout<< arr[t] << " ";
38     }
39 };
40
41
42 int main(){
43     const int maximum = 100;
44
45     int numbers[maximum];
46     srand(time(0));
47     for(int i = 0; i < maximum; i++){
48         numbers[i] = rand() % 100;
49     };
50
51     cout<<"UNSORTED ARRAY"<<endl;
52     display(numbers, maximum);
53
54
55     cout<<"\n";
56
57     cout<<"SORTED ARRAY (Selection Sort)"<<endl;
58     selectionSort(numbers, maximum);
59     display(numbers, maximum);
60
61
62     return 0;
63 }

```

Output

Clear

```

/tmp/APxG3LiEdC.o
UNSORTED ARRAY
38 82 34 50 86 14 92 89 94 67 4 10 16 70 82 26 33 99 12 93 41 13 83 70 70 22 61 18 22
45 44 60 79 31 62 66 97 6 55 92 74 12 54 90 82 37 68 67 88 32 60 29 45 95 51 68 17
64 86 39 9 83 51 41 66 65 7 63 72 14 7 46 26 62 88 60 51 56 27 91 89 40 72 86 35
24 54 53 40 93 44 2 28 96 43 94 61 50 57 85
SORTED ARRAY (Selection Sort)
2 4 6 7 7 9 10 12 12 13 14 14 16 17 18 22 22 24 26 26 27 28 29 31 32 33 34 35 37 38 39
40 40 41 41 43 44 44 45 45 46 50 50 51 51 51 53 54 54 55 56 57 60 60 60 61 61 62
62 63 64 65 66 66 67 67 68 68 70 70 70 72 72 74 79 82 82 82 83 83 85 86 86 86 88
88 89 89 90 91 92 92 93 93 94 94 95 96 97 99

=== Code Execution Successful ===

```

Observation

For the selection sort algorithm it used two functions in order to effectively sort the array. The first function is there for finding the smallest value in the array and getting its position. This is then called in the next function in order to know when to swap the elements in the array. This causes the array to be divided into sorted and unsorted portions, but as long as the position of the small element is tracked the sorting process will be successful just like in the example.

Table 7-3. Selection Sort Algorithm

Code + Console Screenshot

main.cpp



Share

Run

```
1 #include <iostream>
2 #include <cstdlib>
3 #include <ctime>
4
5 using namespace std;
6
7 void display(int arr[], int arr_size){
8     for(int t = 0; t < arr_size; t++){
9         cout<< arr[t] << " ";
10    }
11 };
12
13 void insertionSort(int arr[], int arr_size)
14 {
15     int num_2, key, num_1;
16     for (num_2 = 1; num_2 < arr_size; num_2++) {
17         key = arr[num_2];
18         num_1 = num_2 - 1;
19
20         while (num_1 >= 0 && arr[num_1] > key) {
21             arr[num_1 + 1] = arr[num_1];
22             num_1 = num_1 - 1;
23         }
24         arr[num_1 + 1] = key;
25     }
26 }
27
```

```
28 int main(){
29     const int maximum = 100;
30
31     int numbers[maximum];
32     srand(time(0));
33     for(int i = 0; i < maximum; i++){
34         numbers[i] = rand() % 100;
35     };
36
37     cout<<"UNSORTED ARRAY"<<endl;
38     display(numbers, maximum);
39
40
41     cout<<"\n";
42
43     cout<<"SORTED ARRAY (Insertion Sort)"<<endl;
44     insertionSort(numbers, maximum);
45     display(numbers, maximum);
46
47
48     return 0;
49 }
```

	<div> <div>Output</div> <div>Clear</div> <pre> /tmp/86fi3e3Adz.o UNSORTED ARRAY 14 86 48 44 33 0 73 29 30 7 80 45 78 88 78 29 3 92 24 32 98 13 57 91 38 52 42 44 74 74 64 40 13 12 37 46 65 10 27 47 69 59 93 99 47 71 28 3 16 4 87 14 17 44 5 55 49 99 51 23 25 67 15 38 32 52 36 97 14 64 96 83 23 89 34 23 13 62 26 29 18 65 43 87 10 48 95 59 47 46 34 72 66 49 63 98 2 99 47 68 SORTED ARRAY (Insertion Sort) 0 2 3 3 4 5 7 10 10 12 13 13 13 14 14 14 15 16 17 18 23 23 23 24 25 26 27 28 29 29 29 30 32 32 33 34 34 36 37 38 38 40 42 43 44 44 44 45 46 46 47 47 47 47 48 48 49 49 51 52 52 55 57 59 59 62 63 64 64 65 65 66 67 68 69 71 72 73 74 74 78 78 80 83 86 87 87 88 89 91 92 93 95 96 97 98 98 99 99 99 === Code Execution Successful === </pre> </div>
Observation	<p>In the insertion sort algorithm code, I observed that it uses the index and the value for each process for sorting the array. It starts by getting the second element of the array and the first element in the array. The key is used to compare the values if they should be greater than or less than. The while loop is needed to get the right position to put the key. This process continues until the array is sorted.</p>

Table 7-4. Insertion Sort Algorithm

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

Output Console Showing the Sorted Array	
Pseudo code	<ol style="list-style-type: none"> 1. Start the program. 2. Define a function called `bubbleSort` that takes an array and its size as inputs. <ul style="list-style-type: none"> - Loop through the array starting from the first element to the second-to-last element. - For each element, compare it to the next one. - If the current element is greater than the next one: <ul style="list-style-type: none"> - Swap the two elements. - Continue this process until the array is sorted. 3. Create an array of votes size of 100. 4. Initialize random votes for the candidates (numbers 1 to 5). 5. Print the unsorted votes 6. Call the bubbleSort function to sort the array: 7. print sorted votes 8. End the program.

```

9  template <typename T>
10 void bubbleSort(T arr[], int arr_size) {
11     for (int x = 0; x < arr_size - 1; x++) {
12         for (int y = 0; y < arr_size - 1 - x; y++) {
13             if (arr[y] > arr[y + 1]) {
14                 T temp = arr[y];
15                 arr[y] = arr[y + 1];
16                 arr[y + 1] = temp;
17             }
18         }
19     }
20 }

```

```

22 void display(int arr[], int arr_size) {
23     for (int t = 0; t < arr_size; t++) {
24         cout<< arr[t] << " ";
25     }
26     cout<<endl;
27 }

```

UNSORTED VOTES

```

1 3 3 2 1 3 1 2 1 1 3 4 4 1 1 3 4 2 3 5 1 1 3 2 1 1 5 3 3 5 1 1 2 3 2 5 1 2 1
  3 4 1 3 3 3 3 5 4 4 4 3 4 4 2 3 1 2 4 4 2 4 4 2 2 4 5 1 4 3 4 3 4 4 5 1 1
  2 2 4 2 1 4 1 1 5 5 2 4 3 2 5 3 2 3 5 5 2 5 5 1

```

SORTED VOTES

```

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 2 2 2 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 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5

```

Manual Count

Pseudo Code

1. Start the program.
2. Define a function called `linearSearch`:
 - Input an array, the size of the array, and the value to search for.
 - Set a variable for getting frequency
3. Loop through each element in the array:
 - If the element matches the value being searched for (the candidate number):
 - Increment the frequency by 1.
4. After the loop ends return the frequency.
5. Call the `linearSearch` function for each candidate
6. Print the frequency for each candidate:
7. End the program.

```
29 int linearSearch(int arr[], int arr_size, int find) {  
30     int freq = 0;  
31     for (int x = 0; x < arr_size; x++) {  
32         if (arr[x] == find) {  
33             freq++;  
34         }  
35     }  
36     return freq;  
37 }
```

Count Result of Algorithm

Pseudo Code

1. Start the program.
2. Count the votes for each candidate
3. Print the total votes for each candidate
4. Compare the vote counts
5. Print the winner
 - Print "The winner is Candidate X with Y votes!"
6. End the program.


```

int result1 = linearSearch(numbers, maximum, 1);
int result2 = linearSearch(numbers, maximum, 2);
int result3 = linearSearch(numbers, maximum, 3);
int result4 = linearSearch(numbers, maximum, 4);
int result5 = linearSearch(numbers, maximum, 5);

cout<<"\n";
cout<<"=====TALLY OF VOTES===== "<<endl;

cout<<"Candidate 1| Bo Dalton Capistrano : "<<result1<<" votes"<<endl;
cout<<"Candidate 2| Cornelius Raymon Agustín : "<<result2<<" votes"<<endl;
cout<<"Candidate 3| Deja Jayla Bañaga : "<<result3<<" votes"<<endl;
cout<<"Candidate 4| Lalla Brielle Yabut : "<<result4<<" votes"<<endl;
cout<<"Candidate 5| Franklin Relano Castro : "<<result5<<" votes"<<endl;

int voteCounts[5] = {result1, result2, result3, result4, result5};
int maxVotes = voteCounts[0];
int winnerIndex = 0;

for (int i = 1; i < 5; i++) {
    if (voteCounts[i] > maxVotes) {
        maxVotes = voteCounts[i];
        winnerIndex = i;
    }
}

cout<<"\n";
cout<<"=====ELECTION WINNER===== "<<endl;
switch(winnerIndex) {
    case 0:
        cout<<"Candidate 1| Bo Dalton Capistrano is the winner with "<< maxVotes <<" votes"<<endl;
        break;
    case 1:
        cout<<"Candidate 2| Cornelius Raymon Agustín is the winner with "<< maxVotes <<" votes"<<endl;
        break;
    case 2:
        cout<<"Candidate 3| Deja Jayla Bañaga is the winner with " << maxVotes <<" votes"<<endl;
        break;
    case 3:
        cout<<"Candidate 4| Lalla Brielle Yabut is the winner with " << maxVotes <<" votes"<<endl;
        break;
    case 4:
        cout<<"Candidate 5| Franklin Relano Castro is the winner with "<< maxVotes <<" votes"<<endl;
        break;
}
return 0;
}
|

```

```

=====TALLY OF VOTES=====
Candidate 1| Bo Dalton Capistrano : 24 votes
Candidate 2| Cornelius Raymon Agustín : 19 votes
Candidate 3| Deja Jayla Bañaga : 22 votes
Candidate 4| Lalla Brielle Yabut : 21 votes
Candidate 5| Franklin Relano Castro : 14 votes

=====ELECTION WINNER=====
Candidate 1| Bo Dalton Capistrano is the winner with 24 votes

```

I used the bubble sort technique because I saw it as easiest to grasp and I was able to implement it successfully in the given problem. It gave me an easy time developing the needed functionalities for the problem at hand.

C. Conclusion & Lessons Learned

After this activity I was able to understand the different sorting techniques namely the bubble sort, selection sort, and insertion sort. I was able to grasp their basic structure and implement them into sorting an array of numbers. In selection sort, I learned that it focuses on repeatedly finding the smallest or largest element from the unsorted part of the array and placing it in the correct position. Then in insertion sort, it builds the sorted portion of the array by inserting each element into its correct position within the sorted part. Lastly is the bubble sort. It provides a simple way to think about sorting because it just compares and swaps adjacent elements. After learning the different sorting algorithm I used those to implement in the supplementary activity which is to sort votes and declare the winner by counting the largest instances of the number. By the end of this activity I learned that the different sorting techniques have their own advantages and disadvantages and we must be able to identify when to use the right one for a given situation.