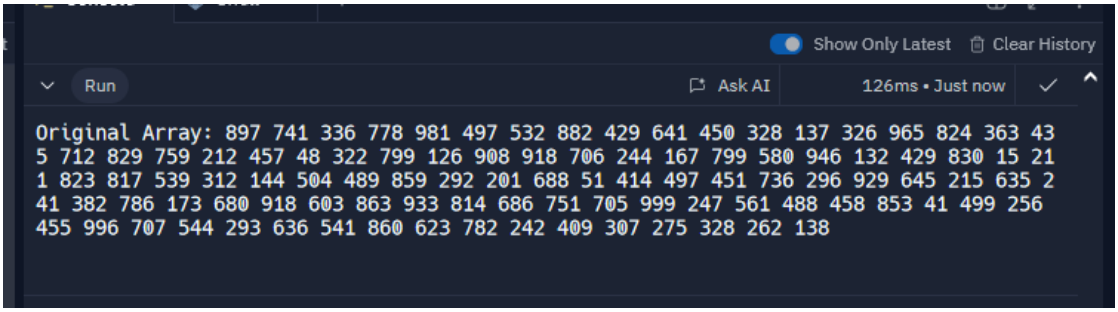
Activity No. <n>	
<Replace with Title>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 10 / 16 / 2024**
Section: CpE21S4	**Date Submitted:**
Name(s): Leoj Jeam B. Tandayu	**Instructor: Ms. Ma. Rizette Sayo**

6. Output

Code + Console Screenshot

```
#include <iostream>
#include <cstdlib>
#include <ctime>

const int size = 100;

void generateRandomArray(int arr[], int size) {
	std::srand(std::time(0));
	for (int i = 0; i < size; i++) {
	arr[i] = std::rand() % 1000;
	}
}

void printArray(int arr[], int size) {
	for (int i = 0; i < size; i++) {
	std::cout << arr[i] << " ";
	}
	std::cout << std::endl;
}

int main() {
	int arr[size];

	// Generate random array
	generateRandomArray(arr, size);

	// Print the original array
	std::cout << "Original Array: ";
	printArray(arr, size);

	return 0;
}
```



Observation	The code generates a random elements which is a 3 digit number for an array with a size of 100.

Table 7-1. Array of Values for Sort Algorithm Testing

Code + Console Screenshot	

```
#include <iostream>

template <typename T>
int Routine_Smallest(T arr[], int K, const int arrSize) {
int position = K;
T smallestElem = arr[K];

for (int j = K + 1; j < arrSize; j++) {
if (arr[j] < smallestElem) {
smallestElem = arr[j];
position = j;
}
}
return position;
}

template <typename T>
void selectionSort(T arr[], const int N) {
for (int i = 0; i < N - 1; i++) {
int POS = Routine_Smallest(arr, i, N);
std::swap(arr[i], arr[POS]);
}
}

int main() {
int data[] = {-5, 72, 0, 33, -9, 100, 57, -6, 105};
int size = sizeof(data) / sizeof(data[0]);

std::cout << "Original Array: ";
for (int i = 0; i < size; ++i) {
std::cout << data[i] << "  ";
```

```
}
std::cout << std::endl;

selectionSort(data, size);

std::cout << "Sorted Array in Ascending Order: ";
for (int i = 0; i < size; ++i) {
std::cout << data[i] << "  ";
}
std::cout << std::endl;

return 0;
}
```

```
Original Array: -5  72  0  33  -9  100  57  -6  105
Sorted Array in Ascending Order: -9  -6  -5  0  33  57  72  100  105
```

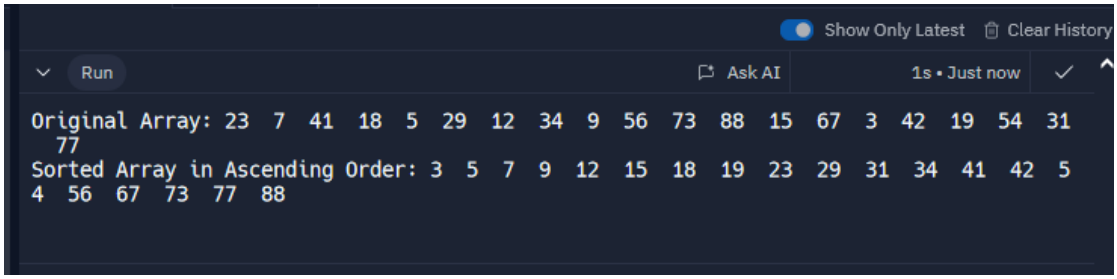
Observation	The code uses bubble sort which sorts all the elements inside an array through looping into ascending form in a way that the largest number is always at the end.

Table 7-2. Bubble Sort Technique

Code + Console Screenshot

```
#include <iostream>

int main() {
int data[] = {23, 7, 41, 18, 5, 29, 12, 34, 9, 56, 73, 88, 15, 67, 3, 42, 19, 54, 31, 77};
int size = sizeof(data) / sizeof(data[0]);

std::cout << "Original Array: ";
for (int i = 0; i < size; ++i) {
std::cout << data[i] << "  ";
}
std::cout << std::endl;

for (int i = 0; i < size - 1; i++) {
int smallestPos = i;
for (int j = i + 1; j < size; j++) {
if (data[j] < data[smallestPos]) {
smallestPos = j;
}
}
std::swap(data[i], data[smallestPos]);
}

std::cout << "Sorted Array in Ascending Order: ";
for (int i = 0; i < size; ++i) {
std::cout << data[i] << "  ";
}
std::cout << std::endl;
```

```
return 0;
}
```



Observation	The selection sorting sorts the numbers in a way that it arranges the numbers by finding the smallest number and placing it in their correct place.

Table 7-3. Selection Sort Algorithm

Code + Console Screenshot	

```
#include <iostream>

int main() {
int data[] = {23, 7, 41, 18, 5, 29, 12, 34, 9, 56, 73, 88, 15, 67, 3, 42, 19, 54, 31, 77};;
int size = sizeof (data) / sizeof(data[0]);

std::cout << "Original Array: ";
for (int i = 0; i < size; ++i) {
std::cout << data[i] << "  ";
}
std::cout << std::endl;

for (int K = 1; K < size; K++) {
int temp = data[K];
int J = K - 1;

while (J >= 0 && data[J] > temp) {
data[J + 1] = data[J];
J--;
}
data[J + 1] = temp;
}

std::cout << "Sorted Array in Ascending Order: ";
for (int i = 0; i < size; ++i) {
std::cout << data[i] << "  ";
}
std::cout << std::endl;

return 0;
}
```

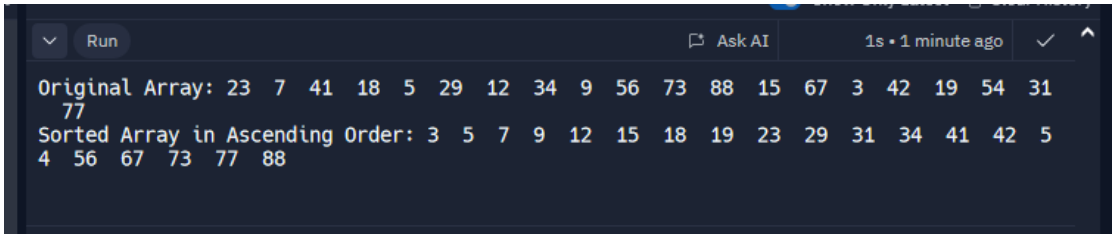
	
Observation	Insertion sort creates a sorted list by adding one item at a time and putting each new item in the right place among the items that are already sorted.

Table 7-4. Insertion Sort Algorithm

7. Supplementary Activity

Source Code:

```
#include <iostream>
#include <ctime>
#include <cstdlib>

using namespace std;

void generateVotes(int votes[], int size) {
    srand(time(0));
    for (int i = 0; i < size; i++) {
        votes[i] = (rand() % 5) + 1;
    }
}

void bubbleSort(int votes[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (votes[j] > votes[j + 1]) {
                int temp = votes[j];
                votes[j] = votes[j + 1];
                votes[j + 1] = temp;
            }
        }
    }
}

void countVotes(int votes[], int size) {
    int count[5] = {0};
    for (int i = 0; i < size; i++) {
        count[votes[i] - 1]++;
    }

    int maxVotes = 0;
    int winningCandidate = 0;
    for (int i = 0; i < 5; i++) {
        if (count[i] > maxVotes) {
            maxVotes = count[i];
            winningCandidate = i + 1;
        }
```

```
    }

    cout << "Vote Count: ";
    for (int i = 0; i < 5; i++) {
        cout << "Candidate " << i + 1 << ": " << count[i] << " ";
    }
    cout << endl;
    cout << "Winning Candidate: Candidate " << winningCandidate << endl;
}

int main() {
    const int size = 100;
    int votes[size];

    generateVotes(votes, size);
    cout << "Votes: ";
    for (int i = 0; i < size; i++) {
        cout << votes[i] << " ";
    }
    cout << endl;

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

    countVotes(votes, size);

    return 0;
}
```

Pseudo Code

```
FUNCTION generateVotes(votes[], size)
    SET seed for random number generator
    FOR i FROM 0 TO size - 1
        SET votes[i] TO a random number between 1 and 5
    END FOR
END FUNCTION

FUNCTION bubbleSort(votes[], size)
    FOR i FROM 0 TO size - 2
        FOR j FROM 0 TO size - i - 2
            IF votes[j] > votes[j + 1]
                SWAP votes[j] and votes[j + 1]
            END IF
        END FOR
    END FOR
END FUNCTION
```

```
FUNCTION countVotes(votes[], size)
   CREATE count array initialized to 0 with size 5
   FOR i FROM 0 TO size - 1
      INCREMENT count[votes[i] - 1]
   END FOR

   SET maxVotes TO 0
   SET winningCandidate TO 0
   FOR i FROM 0 TO 4
      IF count[i] > maxVotes
         SET maxVotes TO count[i]
         SET winningCandidate TO i + 1
      END IF
   END FOR

   PRINT "Vote Count:"
   FOR i FROM 0 TO 4
      PRINT "Candidate i + 1: count[i]"
   END FOR
   PRINT "Winning Candidate: Candidate winningCandidate"
END FUNCTION

FUNCTION main()
   SET size TO 100
   CREATE votes array of size 100

   CALL generateVotes(votes, size)
   PRINT "Votes:"
   FOR each vote in votes
      PRINT vote
   END FOR

   CALL bubbleSort(votes, size)
   PRINT "Sorted Votes:"
   FOR each vote in votes
      PRINT vote
   END FOR

   CALL countVotes(votes, size)
END FUNCTION
```

Output Console Showing Sorted Array	Manual Count	Count Result of Algorithm
Sorted Votes: 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 2 2 2 2 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 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	Sorted Votes: 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 2 2 2 2 2 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 5 5 5 5 5 5 5 5 5 5 5 5	Winning Candidate: Candidate 2

	5 5 5 5 5 5 5 5 5 5 5	

8. Conclusion

In this activity, I have learned the different sorting methods that can be used such as bubble sort, selection sort, and insertion sort. Through the given example of sorting random generated elements of an array, I have learned how these different sorting functions. This is a very helpful knowledge to learn especially for bigger size of datas.

9. Assessment Rubric