Assignment 4.2	
Bubble Sort	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 9/11**
Section: CPE11S1	**Date Submitted: 9/11**
Name(s): Juan Paulo C. Lara	**Instructor: Engr. Jimlord M. Quejado**

6. Output

```
#include <iostream>
using namespace std;
int main()
{
    int scores [10] = {90,85,78,88,92,80,75,80,89,91};
    int n = 10;
    int temp;

    for (int i = 0; i < n - 1; i++){
        for (int j = 0; j < n - i - 1; j++){
            if (scores[j] > scores[j + 1]){
                temp = scores[j];
                scores[j] = scores[j + 1];
                scores[j + 1] = temp;
            }
        }
    }

    cout << "Scores in ascending order: ";
    for (int i = 0; i < n; i++){
        cout << scores [i] << " ";
    }
    return 0;
}
```

```
Scores in ascending order: 75 78 80 80 85 88 89 90 91 92
--------------------------------
Process exited after 0.1905 seconds with return value 0
Press any key to continue . . .
```

During execution of this code, first, line 5 declares an array called scores with 10 indexes ranging from 0 to 9. The numbers fill up these indexes for the next commands to take the data from. Line 6 states n to store the array size so the other commands can refer to it. Line 7 initializes temp as a temporary variable used in swapping. This is needed to prevent one value from being overwritten.

Inside the for loops starting at line 9, the variable i acts as a counter for the amount of attempts made within the loop. The part “i < n - 1” tells the loop to continue right before the last where the largest number stays at the end of the array, hence the term bubble sort. Basically, it scans and swaps values continuously until they are sorted.

For the inner for loop at line 10, the variable j looks up the array from index 0 to 9. The part “n - i - 1” acts similar to the previous subtract by 1 command from the outer for loop, where the check stops before the last value. The variable i

inside this loop calls for the amount of attempts in scanning and sorting the array from 0 to 8, 7, …, 1. As more numbers are organized, the check gets smaller.

The if part tells to swap if the chosen element is larger than the next to move the larger values to the right. For example, since 85 is larger than 70, the elements swap places. For swapping, this is where the variable temp keeps the initial value of score[j] while scores[j] = scores[j + 1] transfer the next value. Then, the last for loop displays the scores sorted in ascending order.

7. Supplementary Activity

8. Conclusion

Concluding this activity, I learned that the for loop can actually be used to arrange data in memory in various ways. This activity also gave me ideas of how filesystems sort out filenames and their metadata in name, date, modified, and size even if it's a very simple model compared to the complexity of a filesystem. Moving back to the discussed code, the multiple for loops allow for checking, scanning, arranging, and displaying values all inside one line per loop. Compared to an if-then model, a for loop can do more complex tasks and take up less lines.

9. Assessment Rubric