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 / 2024
Section: CPE21S4	**Date Submitted:** 10 / 16 / 2024
Name(s): ROALLOS, Jean Gabriel Vincent G.	**Instructor:** Prof. Maria Rizette Sayo

6. Output

Code + Screenshot

main.cpp | sorting.h

```cpp
#include <iostream>
#include <cstdlib>
#include <time.h>
#include "sorting.h"
using namespace std;

const int max_size = 100;

int dataset[max_size];

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

    for (int i = 0; i < max_size; i++)
    {
        cout << dataset[i] << " ";
    }
    return 0;
}
```

input

```
77 77 62 98 32 47 81 53 66 46 32 36 11 19 37 80 23 22 40 88 91 72 54 37 91 19 86 24 44 98 42 74 27
5 72 59 4 53 12 22 0 97 59 63 16 96 43 39 70 83 28 14 8 34 51 99 5 89 75 49 39 18 23 18 75 96 78 31
 1 42 53 53 91 12 17 7 61 12 99 83 96 27 97 56 13 0 7 18 89 82 67 80 52 91 99 79 39 29 10 92

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

Observations

It has generated an array of numbers with randomized values. Adding a “% 100” would limit the recorded elements to under 2 digits.

Code + Screenshot	

```cpp
#include <iostream>
#include <cstdlib>
#include <time.h>
#include "sorting.h"
using namespace std;

const int max_size = 100;

int dataset[max_size];

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

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

    size_t arrSize = sizeof(dataset) / sizeof(dataset[0]);

    bubbleSort(dataset, arrSize);
    cout << "\n\nSorted: " << endl;
    for (size_t i = 0; i < arrSize; i++)
    {
        cout << dataset[i] << " ";
    }

    return 0;
}
```

input

```
6 95 52 39 26 15 80 81 2 99 31 44 88 17 54 61 11 29 18 74 95 52 83 10 81 5 19 60 3 48 10 61 43 15 5
3 21 30 33 54 32 84 86 28 25 55 83 38 18 64 56 92 59 61 75 21 42 32 92 54 87 40 17 49 36 32 2 9 62
87 16 46 72 54 75 49 9 10 87 27 26 95 71 37 8 98 58 2 30 51 57 18 43 74 19 31 6 21 41 68 60

Sorted:
99 98 95 95 95 92 92 88 87 87 87 86 84 83 83 81 81 80 75 75 74 74 72 71 68 64 62 61 61 61 60 60 59
58 57 56 55 54 54 54 54 53 52 52 51 49 49 48 46 44 43 43 42 41 40 39 38 37 36 33 32 32 32 31 31 30
30 29 28 27 26 26 25 21 21 21 19 19 18 18 18 17 17 16 15 15 11 10 10 10 9 9 8 6 6 5 3 2 2 2

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

Observations	The sorting algorithm works as intended and has arranged the randomized values from the array into a descending order.

Code + Screenshot	main.cpp sorting.h
	``
	#include <iostream>
	#include <cstdlib>
	#include <time.h>
	#include "sorting.h"
	using namespace std;
	
	const int max_size = 100;
	
	int dataset[max_size];
	
	int main()
	{
	srand(time(0));
	for (int i = 0; i < max_size; i++)
	{
	dataset[i] = rand() % 100;
	}
	
	for (int i = 0; i < max_size; i++)
	{
	cout << dataset[i] << " ";
	}
	
	size_t arrSize = sizeof(dataset) / sizeof(dataset[0]);
	
	//bubbleSort(dataset, arrSize);
	selectionSort(dataset, max_size);
	
	cout << "\n\nSorted: " << endl;
	for (size_t i = 0; i < arrSize; i++)
	{
	cout << dataset[i] << " ";
	}
	
	return 0;
	``
	input
	``
	6 10 75 42 89 22 28 78 86 69 88 26 34 63 28 11 55 89 58 66 61 15 2 17 1 40 90 68 97 21 86 3 32 13 9
	8 21 35 78 52 21 48 92 0 34 55 28 98 62 69 8 29 82 23 83 0 77 24 42 45 73 64 83 28 96 96 26 69 83 5
	21 56 5 14 8 91 21 88 89 84 57 98 13 92 73 48 92 50 24 86 47 97 50 82 26 98 78 4 20 13 61
	
	Sorted:
	0 0 1 2 3 4 5 5 6 8 8 10 11 13 13 13 14 15 17 20 21 21 21 21 21 22 23 24 24 26 26 26 28 28 28 28 29
	32 34 34 35 40 42 42 45 47 48 48 50 50 52 55 55 56 57 58 61 61 62 63 64 66 68 69 69 69 73 73 75 77
	78 78 78 82 82 83 83 83 84 86 86 86 88 88 89 89 89 90 91 92 92 92 96 96 97 97 98 98 98 98
	
	...Program finished with exit code 0
	Press ENTER to exit console.
	``

Observations	The selection sort algorithm outputted a sorted array of the randomized values into an ascending order.

Code + Screenshot	main.cpp sorting.h
	``
	#include <iostream>
	#include <cstdlib>
	#include <time.h>
	#include "sorting.h"
	using namespace std;
	
	const int max_size = 100;
	int dataset[max_size];
	
	int main()
	{
	srand(time(0));
	for (int i = 0; i < max_size; i++)
	{
	dataset[i] = rand() % 100;
	}
	
	for (int i = 0; i < max_size; i++)
	{
	cout << dataset[i] << " ";
	}
	
	size_t arrSize = sizeof(dataset) / sizeof(dataset[0]);
	
	//bubbleSort(dataset, arrSize);
	//selectionSort(dataset, max_size);
	insertionSort(dataset, max_size);
	
	cout << "\n\nSorted: " << endl;
	for (size_t i = 0; i < arrSize; i++)
	{
	cout << dataset[i] << " ";
	}
	
	return 0;
	``
	input
	``
	59 80 45 6 46 16 87 26 81 42 93 37 38 5 93 73 10 71 39 92 61 71 57 84 35 41 77 87 79 35 54 90 67
	99 48 13 68 87 92 49 30 37 39 68 94 32 94 4 3 33 48 64 57 58 49 92 99 78 79 30 13 86 72 33 37 20
	98 5 8 90 7 90 27 46 10 73 78 4 77 33 90 26 98 47 36 99 91 35 77 70 17 42 56 89 75 46 9 26 51 69
	
	Sorted:
	3 4 4 5 5 6 7 8 9 10 10 13 13 16 17 20 26 26 26 27 30 30 32 33 33 33 35 35 35 36 37 37 37 38 39 3
	9 41 42 42 45 46 46 46 47 48 48 49 49 51 54 56 57 57 58 59 61 64 67 68 68 69 70 71 71 72 73 73 75
	77 77 77 78 78 79 79 80 81 84 86 87 87 87 89 90 90 90 90 91 92 92 92 93 93 94 94 98 98 99 99 99
	
	...Program finished with exit code 0
	Press ENTER to exit console.
	``
	(rendered as a dark console window)

Observations	Same with the selection sort algorithm, the output was a sorted array of randomized values in ascending order.

7. Supplementary Activity

main.cpp | sorting.h | supplementary-

```cpp
int main()
{
    int arr[max_size];
    int numVotes[candidates] = {};

    voteGen(arr, max_size);

    cout << "Unsorted array of votes: \n";
    outputLoop(arr, max_size);

    insertionSort(arr, max_size);

    cout << "\n Sorted array of votes: \n";
    outputLoop(arr, max_size);

    voteCount(arr, max_size, numVotes);

    cout << "Vote Tally: \n";
    for (int i = 0; i < candidates; i++)
    {
        cout << "Candidate #" << i + 1 << ": " << numVotes[i] << " votes" << endl;
    }

    int winner = showResults(numVotes, candidates);
    cout << "Winning candidate: Candidate #" << winner;

    return 0;
```

input

```
Unsorted array of votes:
3 2 3 4 5 5 2 5 2 4 4 1 1 5 4 3 2 1 5 3 1 3 4 5 1 4 1 5 2 2 5 4 4 2 5 3 3 3 4 2 3 3 4 3 4 2 1 1 3 5 5 5 4 5 4 4 1 1 5
2 3 4 2 1 2 1 3 1 1 4 2 3 1 5 1 4 4 1 4 1 2 5 5 5 5 1 5 2 1 1 3 3 1 4 3 4 2 3 4 2

 Sorted array of 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 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 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
Vote Tally:
Candidate #1: 22 votes
Candidate #2: 17 votes
Candidate #3: 19 votes
Candidate #4: 22 votes
Candidate #5: 20 votes
Winning candidate: Candidate #1

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

Output Console Showing Sorted Array	Manual Count	Output of Algorithm
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	22	22
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	17	17
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3	19	19
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4	22	22
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5	20	20

The code has successfully generated votes, sorted the values in ascending order, and tallied the count of each element. Therefore we can conclude that the code works as intended.

8. Conclusion
After the activity, I have been more familiarized with the various sorting techniques used in programming. I have been curious on how these work and how or what would make them differ from each other
9. Assessment Rubric