Seatwork 7.1

Using Sorting Algorithms

Course Code: CPE010	**Program:** BSCPE
Course Title: Data Structures and Algorithms	**Date Performed:** September 16, 2025
Section: CPE21S4	**Date Submitted:** September 16, 2025
Name: Bautista, Mariela S.	**Instructor:** Engr. Quejado

Objective(s):

Objectives: To create a program in C++ using sorting algorithms.
1. To be familiarized with the sorting algorithms.
2. To be able to different the types of sorting algorithms.
3. To be able to create a program with sorting algorithms.

Output:

1. What is sorting algorithm?
   A sorting algorithm is the rearranging of the given array or list of elements in an order. There are so many techniques how to operate this sorting algorithm such as; ascending wherein the order of the elements should be in increasing numbers while the other one, which is Descending order, the elements should be in a decreasing order. Sorting algorithm are very useful for making the arrays or elements much easier to implement and read by the system. In this way, less error and the sorting time is reduced in a significant amount of time. To put it simply, this a step-by-step arrangement in order either into smallest to biggest order or vice versa.

2. Where can sorting algorithms be used?
   Sorting algorithms are used whenever we need an organized data;
   - Organizing things in order whether it's an important document, files, certificates and such. This is convenient so whenever we need it and we start looking for it, we can easily find it wherever we stored it.
   - Scores of the students on a quiz from highest to lowest. Or from students who needs more improvements to those students who's doing well.
   - Sorting the contact lists.
   - Alphabetical order of the names of your students.
   - Necessity kit for when we need it, we can easily find and use it.

3. Explain the different types of sorting problem.
   The different types of sorting algorithms that are discussed in today's lesson are;
   Insertion sorting, I think of insertion sort like how I sort cards in my hand. I take one number and place it in the right spot among the ones already sorted. If something's bigger, it just shifts over to make space. It works really well for small lists or ones that are already almost sorted. To me, it feels natural and a bit faster than selection sort sometimes.

   Selection sorting, For me, selection sort feels like picking the smallest number and putting it in front. I just keep doing that until the whole list is sorted. It's simple to understand, but I know it can be slow if the list is long. Still, it's an easy way to see how sorting works. That's why I find it straightforward to use.

Bubble sorting, for bubble sort, I imagine swapping neighbors if they're in the wrong order. The biggest numbers move to the end, like bubbles rising up. I just repeat that process until nothing needs to be swapped. It's super easy for me to follow, even if it's kinda slow for big lists. That's why I think it's the most basic sorting method to learn first.

4. Give sample programs in C++ that uses sorting algorithms, specifically selection sort, insertion sort, and bubble sort. Explain how the programs work.

Selection Sort Code:	main.cpp
	1 #include <iostream>
	2 using namespace std;
	3 void Selection_Sort(int arr[], int n)
	4 {
	5 for(int i = 0; i < n - 1; ++i)
	6 {
	7 int min_index = i;
	8 for(int j = i + 1; j < n; ++j)
	9 {
	10 if(arr[j] < arr[min_index])
	11 min_index = j;
	12 }
	13 swap(arr[i], arr[min_index]);
	14 }
	15 }
	16 int main()
	17 {
	18 int n = 5;
	19 int arr[5] = {2, 0, 1, 4, 3};
	20 Selection_Sort(arr, n);
	21 cout<<"The Sorted Array by using Selection Sort is : ";
	22 for(int i = 0; i < n; ++i)
	23 cout<<arr[i]<<" ";
	24 return 0;
	25 }

Output:	The Sorted Array by using Selection Sort is : 0 1 2 3 4 ...Program finished with exit code 0 Press ENTER to exit console.
Analysis:	When I tried this program, I see how selection sort works by always picking the smallest number to largest number and putting it into the right place. It felt like scanning a list trying to find the tiniest number, and moving it forward each time. So for me, this is pretty easy to comprehend since I can clearly see the list getting sorted step-by-step.
References :	Selection Sort VS Bubble Sort - GeeksforGeeks

Insertion Sort Code:	main.cpp
	1 // Insertion sort in C++
	2
	3 #include <iostream>
	4 using namespace std;
	5
	6 // Function to print an array
	7 void printArray(int array[], int size) {
	8 for (int i = 0; i < size; i++) {
	9 cout << array[i] << " ";
	10 }
	11 cout << endl;
	12 }
	13
	14 void insertionSort(int array[], int size) {
	15 for (int step = 1; step < size; step++) {
	16 int key = array[step];
	17 int j = step - 1;
	18
	19 // Compare key with each element on the left of it until an element smaller than
	20 // it is found.
	21 // For descending order, change key<array[j] to key>array[j].
	22 while (j >=0 && key < array[j]) {
	23 array[j + 1] = array[j];
	24 --j;
	25 }
	26 array[j + 1] = key;
	27 }
	28 }
	29
	30 // Driver code
	31 int main() {
	32 int data[] = {9, 5, 1, 4, 3};
	33 int size = sizeof(data) / sizeof(data[0]);
	34 insertionSort(data, size);
	35 cout << "Sorted array in ascending order:\n";
	36 printArray(data, size);
	37 }

Output:	Sorted array in ascending order: 1 3 4 5 9 ...Program finished with exit code 0 Press ENTER to exit console.
Analysis:	I can see how insertion sorting array works by step-by-step, like im sliding into the right spot one at a time. When I ran it, I noticed the numbers slowly ascends higher, which felt like arranging cards in order. For me, its simple to follow because I can clearly see how each number find its proper place.
References :	Insertion Sort (With Code in Python/C++/Java/C)

Bubble Sorting Code:	main.cpp
	1 #include <iostream>
	2 using namespace std;
	3 void Bubble_Sort(int arr[], int n)
	4 {
	5 for(int i = 1; i < n; ++i)
	6 {
	7 for(int j = 0; j <= (n - i - 1); ++j)
	8 {
	9 if(arr[j] > arr[j + 1])
	10 swap(arr[j], arr[j + 1]);
	11 }
	12 }
	13 }
	14
	15 int main()
	16 {
	17 int n = 5;
	18 int arr[5] = {2, 0, 1, 4, 3};
	19 Bubble_Sort(arr, n);
	20 cout<<"The Sorted Array by using Bubble Sort is : ";
	21 for(int i = 0; i < n; ++i)
	22 cout<<arr[i]<<" ";
	23 return 0;
	24 }

Output:	The Sorted Array by using Bubble Sort is : 0 1 2 3 4 ...Program finished with exit code 0 Press ENTER to exit console.
Analysis:	When I ran this program, I see how bubble sorts keep neighbouring orders if in not order until we get the right order of the elements. Its like the bigger numbers slowly move to the end after each pass, just like bubbles rising up. For me, this is the easiest to comprehend because I can see the numbrs getting sorted little by little.
References:	Selection Sort VS Bubble Sort - GeeksforGeeks

Conclusion:

What ive learned today in doing these sorting programs is, I got to see how each method works differently but still ends up putting the numbers in order. At first, it was a bit tricky, but running the codes helped me actually understand the process, like watching the numbers move step by step. I realized that practice makes it easier to follow, and I feel more confident now in using these algorithms. For improvement, I think I just need to practice more with bigger lists and try combining what I learned in one program. Overall, the experience made sorting clearer and more natural for me.

Assessment Rubric:

Reflection / Essay Rubric

Criteria	Ratings				Pts
Focus on Assigned Topic	**4 pts** **Exemplary** Entire essay is related to the assigned topic and allows the reader to understand much more about the topic.	**3 pts** **Proficient** Most of the essay is related to the assigned topic. The essay wanders off at one point, but the reader can still learn something about the topic.	**2 pts** **Acceptable** Some of the essay is related to the assigned topic, but a reader does not learn much about the topic.	**1 pts** **Beginner** No attempt has been made to relate the essay to the assigned topic.	4 pts
Reflection of Personal Learning	**4 pts** **Exemplary** Shows great depth of knowledge and learning, reveals feelings and thoughts, abstract ideas reflected through use of specific details.	**3 pts** **Proficient** Relates learning with research and project, personal and general reflections included, uses concrete language.	**2 pts** **Acceptable** Does not go deeply into the reflection of learning, generalizations and limited insight, uses some detail.	**1 pts** **Beginner** Little or no explanation or reflection on learning, no or few details to support reflection.	4 pts
Mechanics	**4 pts** **Exemplary** No grammatical spelling or punctuation errors.	**3 pts** **Proficient** Almost no grammatical, spelling, or punctuation errors.	**2 pts** **Acceptable** A few grammatical spelling, or punctuation errors.	**1 pts** **Beginner** Many grammatical spelling, or punctuation errors.	4 pts
Organization	**4 pts** **Exemplary** The essay is very well organized. One idea or scene follows another in a logical sequence with clear transitions.	**3 pts** **Proficient** The essay is pretty well organized. One idea may seem out of place. Clear transitions are used.	**2 pts** **Acceptable** The essay is a little hard to follow. Paragraphs are unclear. The transitions are sometimes not clear.	**1 pts** **Beginner** Ideas seem to be randomly arranged. No effort at paragraph organization.	4 pts
Conclusion	**4 pts** **Exemplary** The conclusion is engaging and restates personal learning.	**3 pts** **Proficient** The conclusion restates the learning.	**2 pts** **Acceptable** The conclusion does not adequately restate the learning.	**1 pts** **Beginner** Incomplete and/or unfocused.	4 pts

Total Points: 20