Final (100 Points)

For this final, you will ask the user to enter the course dept, course num and course name. From there, you will ask the user to enter the name of the assignment. After asking the user to enter the name of the assignment, then you will ask the user to enter points possible of the assignment. From there, you will go ahead and enter the number of students that will be scored. Then, you will create a dynamic array to inject the number of elements that will be used to record the scores by using this code: int *arr = new int(n); Then, using a loop, you will entered the score for each student, which the loop should go on for the specified number of times as you have mentioned in the previous question. From there, you will create a function that will be used to sort the scores in numerical order in ascending format by using Recursive Bubble Sort. In that function, you will make it a recursive function, where the function needs to be called for several times until all elements of the array are sorted in alphabetical order.

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

Example:
First Pass:
(**5 1** 4 2 8) –> (**1 5** 4 2 8), Here, algorithm compares the first two elements, and swaps since 5 > 1.
(1 **5 4** 2 8) –> (1 **4 5** 2 8), Swap since 5 > 4
(1 4 **5 2** 8) –> (1 4 **2 5** 8), Swap since 5 > 2
(1 4 2 **5 8**) –> (1 4 2 **5 8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:
(**1 4** 2 5 8) –> (**1 4** 2 5 8)
(1 **4 2** 5 8) –> (1 **2 4** 5 8), Swap since 4 > 2
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:
(**1 2** 4 5 8) –> (**1 2** 4 5 8)
(1 **2 4** 5 8) –> (1 **2 4** 5 8)
(1 2 **4 5** 8) –> (1 2 **4 5** 8)
(1 2 4 **5 8**) –> (1 2 4 **5 8**)

Following is iterative Bubble sort algorithm :

```
// Iterative Bubble Sort
bubbleSort(arr[], n)
{
  for (i = 0; i < n-1; i++)

    // Last i elements are already in place
    for (j = 0; j < n-i-1; j++)
    {
       if(arr[j] > arr[j+1])
          swap(arr[j], arr[j+1]);
    }
}
```

How to implement it recursively?
Recursive Bubble Sort has no performance/implementation advantages but can be a good question to check one's understanding of Bubble Sort and recursion.
If we take a closer look at Bubble Sort algorithm, we can notice that in first pass, we move largest element to end (Assuming sorting in increasing order). In second pass, we move second largest element to second last position and so on.
Recursion Idea.

- Base Case: If array size is 1, return.
- Do One Pass of normal Bubble Sort. This pass fixes last element of current subarray.
- Recur for all elements except last of current subarray.

As you are building this algorithm, be sure to initialize the following library in your program: #include <bits/stdc++.h>

From there, you will go ahead and calculate the average score, minimum and maximum score using the elements of the array. The average score, minimum and maximum score should each have a function that should contain source code for each of its functions.

Then, you will output the course dept followed by space, then the course num followed by colon and finally, the course name. From there, you will output all the scores that was entered in the program earlier but in ascending order. Then, you will output the average score, minimum score and maximum score of the assignment.

Sample Output:
Welcome to Grade Calculator!
Enter the course dept: _____
Enter the course number: _____
Enter the course name: _____

Enter the number of students that will be scored: _____
Enter points possible: _____

//Based on the number of students that was entered, you will run the loop for that amount of times
Enter the score: _____

//After the loop, you will take the element of the array, pass it to the function to do Recursive Sort Algorithm as a recursive function

//Calculate the average score using the following formula
double average = ((total score, add all elements of array together)/(number of students * points possible)) * 100

//Calculate the lowest score

//Calculate the highest score

Course: CMPS 367: Object Oriented Language C++
Scores: 10 20 30 40 50 60 70 75 80 85 90 95 100

Average: _____%
Lowest Score: 10
Highest Score: 100

Thank you for using the grade calculator!

Deliverables:

- A zip file uploaded to Github consisting of the following items:
  - C++ source code
  - Comments on your source code
  - Readme.md in Github that will contain the following information:
    - Overview of your app (What is it, who is the intended audience, problem statement)
    - Features of the app
  - Screenshots of your output to demonstrate program functionality