Homework 3

100 Points

Pointers and Dynamic Allocation of Memory

22B_H3A_InsertSort.cpp (pointers, arrays, and sorting)
22B_H3B_Errors.cpp (find and fix errors: dynamic memory allocation)
22B_H3C_Survey.cpp **Project: Video Games Statistics** (see next pages)

Grading

Program 3A	– 10	
Program 3B	– 20	
Program 3C		
1. Read input file's name	– 5	
2. Get data from file (including dynamically allocating the array)	– 20	
3. Insertion sort	– 5	// reuse code: 3A
4. Write sorted array to screen	– 5	// reuse code: 3A
5. Write sorted array to file	– 10	// change code: 3A (output file)
6. Average	– 10	
7. Display statistics	– 10	
Self Assessment Report	– 5	

Run each program as required and save the output at the end of the source file as a comment. Compress the source file, input and output files (if any), and the report, and upload the compressed file: 22B_LastName_FirstName_H3.zip

Note: Three ways to display an array:

```
// A. Use an index
for( i = 0; i < size; i++ )
{
     cout << ary[i] << " ";
}
// B. Use an index and pointer arithmetic: NEVER USE THIS STYLE!
for( i = 0; i < size; i++ )
{
     cout << *(ary + i) << " ";
}
// C. Use a pointer // ← This is the required style for Project C
for( pW = ary, pLast = ary + size - 1; pW <= pLast; pW++ )
{
     cout << *pw << " ";
}
```

Next Page

Project: Video Games Statistics

Write a program that can be used to gather statistical data about the number of hours per week college students play video games. The program should perform the following steps:

1). Read data from the input file into a dynamically allocated array. The first number in the input file, **n**, represents the number of students that were surveyed. First read this number then use it to dynamically allocate an array of **n** integers. On the next **n** lines, there is an integer representing the number of hours per week each student played video games. Here is an example:

```
5       // There are 5 students
9       // 9 hours per week – student 1
4       // 4 hours per week – student 2
10      // 10 hours per week – student 3
4       // 4 hours per week – student 4
7       // 7 hours per week – student 5
```

Read the rest of the numbers from the file into the dynamically allocated array.

2). Sort the array in ascending order:

```
4 4 7 9 10
```

3). Write the sorted array to a file, one number per line, including the number of students on the first line (same format as the input file)

4). Calculate the average of the numbers in the array without the lowest and the highest values:

```
(4 + 7 + 9) / 3 = 6.6
```

5). Display the number of students, the average, the lowest and highest values in a readable format of your choice (screen output).

6). Finally, release the memory and terminate the program.

Run the program using the following input files:

1. Input file name: **videogamesurvey.txt**
2. Input file name: **gamestats.txt**
3. Input file name: **survey.txt**

Next Page

videogamesurvey.txt

```
5
9
4
10
4
7
```

gamestats.txt

```
10
4
1
2
0
3
2
1
3
0
5
```

survey.txt, with the following data:

```
27
2
8
11
1
2
8
0
3
1
2
9
2
0
3
1
8
9
1
9
0
8
3
1
7
8
12
2
```