CSC 122 – Data Structures
Homework #1 – Introduction to C++

Due: Tuesday 2/4 at 8:00 AM, via Moodle

In this assignment, you'll have three functions to write. You'll also need a main function that calls each of them. So, start by making a Visual Studio project (*lastname*-hw1) and creating a .cpp file to contain your code. Put the following main in your file:

```
int main() {
      cout << "---------- Exercise 1 ----------" << endl;
      hey();

      cout << "---------- Exercise 2 ----------" << endl;
      arrayStats();

      cout << "---------- Exercise 3 ----------" << endl;
      testLoops();

      return 0;
}
```

That way, you can write the functions for these exercises above the main, and everything will get called nicely. (You may need to temporarily comment-out some of the function calls while you're working on the assignment, though.)

In the future, I'll ask you to just write this kind of main yourself, but I wanted to give it to you this first time as an example.

Style

On this and all assignments, please be certain to follow some standard style guidelines:

- All variables, functions, methods, classes, etc. should have descriptive names.
- All code should be indented properly.
- Use consistent spacing. For example, I don't mind if you do: `x = y + z` or `x=y+z`, but whatever you do, please do it consistently.
- Use blank lines now and then to separate your code into logical chunks.

The bottom line is, please take pride in your code and make it look professional. Particularly messy or inconsistent style may lead to lost points. If you have any questions about this, please let me know!

---

1. Write a C++ function called `hey` that prompts the user to enter a name, greets the user by name, and then prints a goodbye message. More specifically, if you type in "Steve" for example:

   ```
   Please enter your name: Steve
   Hello, Steve!
   See ya.
   ```

   The blue represents the part that the user would type in, though when you type in your program, it won't be blue; I just did that here to differentiate it.

Of course, your program should work correctly for any name, not just Steve.

Please don't forget the ! after the name in the second line.

---

2. In the same Visual Studio project, in the same file, write another function named `arrayStats`. It should ask the user to type 5 doubles, one at a time. Each double should be stored at the next spot in an array. Then, print the average value, and finally, each individual value minus the average. (This is what would be needed to compute the statistical value known as a *standard deviation*, but just to keep things simpler, we won't actually compute the standard deviation here.)

   For example:

```
Please enter 5 numbers:
3
4
5
2.5
1
The average is: 3.1
Here are the differences from the average:
For the value at index 0, subtracting the average gives: -0.1
For the value at index 1, subtracting the average gives: 0.9
For the value at index 2, subtracting the average gives: 1.9
For the value at index 3, subtracting the average gives: -0.6
For the value at index 4, subtracting the average gives: -2.1
```

   Again, as I'll do throughout the semester, the blue represents the stuff the user types in. Everything else should be generated by the program based on what is typed in. Of course, your code should generate the correct results (average and differences) for whatever numbers are typed in.

   Please make sure your output is formatted the same as mine above – the same sentences, spacing, line-breaks, etc.

   To receive credit, when you're dealing with the elements of the array (reading numbers in, computing the sum for the average, or computing differences), use a loop. Don't just hardcode nums[0]+nums[1]+... or copy the same kind of line five times or something.

---

3. Write a function called testLoops. It should first ask the user to enter an integer n. Then write a while loop, a do-while loop, and a for loop, each of which accomplishes the same thing: printing all even integers in (0, n] (that is, excluding 0, including n), in *descending* order. *Do this by maintaining a single variable i, decrementing and testing it as is done in each type of loop.*

   Your output should be exactly as the examples below (including little details like how everything is lined up nicely):

   Example 1:

```
Enter n: 20
   While: 20 18 16 14 12 10 8 6 4 2
Do-While: 20 18 16 14 12 10 8 6 4 2
     For: 20 18 16 14 12 10 8 6 4 2
```

Example 2:
```
Enter n: 23
   While: 22 20 18 16 14 12 10 8 6 4 2
Do-While: 22 20 18 16 14 12 10 8 6 4 2
     For: 22 20 18 16 14 12 10 8 6 4 2
```

What happens to your code if you enter a negative value for n? If it goes into an infinite loop, fix it to prevent this from happening.

Consider what happens if the user enters 0 for n. This demonstrates under what condition you might want to use one type of loop instead of another.

Note: Don't forget that a do-while requires a semicolon at the end (after the condition).

Turning In Your Work

As with every assignment this semester, please follow the instructions from hw0, exercises 2 and 3, for turning in this work to Moodle.