CSCE 240 – Programming Assignment Four

Due: 11:59pm on Friday, October 25

Purpose – Implement two classes

Create a *Height* class that holds the value and units of a height in private double and string data members, respectively.

The class must include the following public member functions:

A *SetValue* function that has a double as its parameter and sets the value data member to the argument's value as long as the argument is non-negative, leaving the height unchanged if the argument is negative. The function should return true if the value is set to the function's argument, and false otherwise.

A *GetValue* function that returns a copy of the value data member.

A *SetUnits* function that takes a string as its parameter and sets the unit data member to the string as long as the string is inches, feet, centimeters, or meters. The function should return true if the unit is set to the function's argument, and false otherwise.

A *GetUnits* function that returns a copy of the unit data member.

A *ConvertUnits* function that takes a string parameter for units to convert the Height to. If the argument is valid, the object's value and units should be updated to the equivalent height in the new units. For example, if Height object h has a value of 2 and units of "feet", then after the function call h.ConvertUnits("inches"); h should have a value of 24 and units of "inches". If the argument sent to the function is not "inches" "feet" "centimeters" or "meters" then the function should leave the object unchanged.

A constructor that takes a double for the height's value and a string for the height's units as parameters. The parameters should have default arguments of 0 and "feet", respectively.

Overload the << operator to output a Height object in the format: value units. For example,

```
Height h(8, "feet");
cout << h;  // should output "8 feet" to the standard output device
```

Review initial tests for constructor, mutators, and accessors in *testheight1.cc*. Review initial tests for the *ConvertUnits* function in *testheight2.cc*. And review tests for the stream insertion operator in *testheight3.cc*. If you place all of the attached files in the same directory, you can run the initial tests with the commands

```
make testheight1
make testheight2
make testheight3
```

You are strongly encouraged to create more rigorous tests.

Create an *HeightRange* class that has a smallest *Height* object and a largest *Height* object as private data members.

The class must include the following public member functions:

A *SetShortest* function that takes a const Height& as an argument. The function should update the smallest Height for the range if the value of the argument is not larger than the current value of the largest Height for the range. If the value of the argument is larger than the current value of the largest Height for the range, the function should leave the smallest Height data member unchanged.

A G*etShortest* function that returns a copy of the value of the data member (a Height).

A *SetTallest* function that takes a const Height& as an argument. The function should update the largest Height for the range if the value of the argument is not smaller than the current value of the smallest Height for the range. If the value of the argument is smaller than the current value of the smallest Height for the range, the function should leave the largest Height data member unchanged.

A G*etTallest* function that returns a copy of the value of the data member (a Height).

A default constructor that initializes the shortest and tallest heights to 0 feet.

A constructor that takes two Heights (as constant reference parameters) as arguments. The constructor should initialize the data member for the shortest height for the range to the smaller of the two arguments, and it should set the data member for the tallest height for the range to the larger of the two arguments.

An *InRange* function that takes a const Height& as the first parameter, and a bool (that defaults to true) as the second parameter. The function will return true if the Height argument is within the HeightRange, and false if the Height argument is not within the weight range. The second argument determines whether or not the endpoints of the HeightRange should (true) or should not (false) be considered in range. For example,

```
Height h1(5, "feet"), h2(7, "feet");
HeightRange hr(h1, h2);
hr.InRange(h1);  // should return true since the default argument of
                 // true means the endpoints of the range are in range
hr.InRange(h1, false);  // should return false since the second
                        // argument of false means the endpoints of
                        // the range are not in range.
```

A *Width* function that returns a Height representing the difference between the tallest and shortest Height in the range. The units of the returned value should match the units of the data member for the tallest height in the range.

Review initial tests for constructor and accessors in *testheightrange1.cc*. Review initial tests for the mutator functions in *testheightrange2.cc*. Review initial tests for the *InRange* function in *testheightrange3.cc*. And review tests for the *Width* function in *testheightrange4.cc*. If you place all of the attached files in the same directory, you can run the initial tests with the commands

make testheightrange1
make testheightrange2
make testheightrange3
make testheightrange4

You are strongly encouraged to create more rigorous tests.

Specifications

- Add all code for the definition of the *Height* class in a header file named *height.h*
- Include all of the necessary code for the *Height* class, including the implementation all of the public member functions and the overloaded stream insertion operator, in a source file named *height.cc*
- Add all code for the definition of the *HeightRange* class in a header file named *heightrange.h*
- Include all of the necessary code for the *HeightRange* class, including the implementation all of the public member functions, in a source file named *heightrange.cc*
- You will submit a zip file (only a zip file will be accepted) containing *height.h, height.cc, heightrange.h* and *heightrange.cc* to the assignment in Blackboard.
- Source files must compile and run on a computer of the instructor's choosing in the Linux lab (see your course syllabus for additional details).

Grade Breakdown

Style *height.h*: 0.25 points

Style *height.cc*: 0.25 points

Style *heightrange.h*: 0.25 points

Style *heightrange.cc*: 0.25 points

Documentation: 1 point

Clean compilation of *height.cc*: 0.5 points

Clean compilation of *heightrange.cc*: 0.5 points

Height class passes instructor's modified *testheight1.cc* tests: 1 point

Height class passes instructor's modified *testheight2.cc* tests: 1 point

Height class passes instructor's modified *testheight3.cc* tests: 1 point

HeightRange class passes instructor's modified *testheightrange1.cc* tests: 1 point

HeightRange class passes instructor's modified *testheightrange2.cc* tests: 1 point

HeightRange class passes instructor's modified *testheightrange3.cc* tests: 1 point

HeightRange class passes instructor's modified *testheightrange4.cc* tests: 1 point

The penalty for late program submissions is 10% per day, with no submission accepted after 3 days.