Lab 5 - Distance class

Goals

By the end of this lab you should:

- be more familiar with declaring and using your own ADT (class).
- know how to declare and implement an overloaded operator function

Distance class

Distance represents a length, consisting of an unsigned integer `feet` and a double `inches`. This length should always be a positive value. Inches should never be greater than or equal to 12.

The Class Specification

Your Distance class will provide to the user the following functions:

- a default constructor that constructs a length of 0 (0 feet and 0 inches).
- two "initializing" constructors:
    - `Distance(unsigned ft, double in)` that constructs a length of `ft` feet and `in` inches, unless `in` >= 12.0, in which case the values of `feet` and `inches` are adjusted accordingly.
    - `Distance(double in)` that constructs a length of `in` inches, correctly converted to `feet` and `inches`.

    Note: A Distance may be never be negative: if a negative value is found, convert it to a positive value.
- a private helper function `init()` used by the constructors to properly

initialize a positive length and convert inches >= 12 to feet.

- a member function `convertToInches()` that returns the distance in all inches – does **not** change the state of the object.

- overloaded operators for:

    - add (`+`): the sum of two Distances;

    - subtract(`-`): the difference between two Distances *(the result must always be a proper Distance object, i.e. a positive distance, no matter what order the operands are in)*;

- display function that outputs to standard output (cout) the Distance in the format: `feet' inches" (e.g.: 3' 4.25")`

Required Class Interface

Your class declaration must look like this **EXACTLY!**

```
class Distance
{
  private:
    unsigned feet;
    double inches;
  public:
    Distance();
    Distance(unsigned, double);
    Distance(double);
    double convertToInches() const;
    const Distance operator+(const Distance &) const;
    const Distance operator-(const Distance &) const;
    void display() const;
  private:
    void init();
};
```

Required Files

You must separate your class into the files Distance.h (interface) and Distance.cpp (implementation). Name the test harness file main.cpp.

Testing your functions

You can use the following main function to help with testing your Distance class. You should comment out tests of the functions you have not implemented yet. I recommend you start with the constructors and display function. Then, add one function and test it before moving on to the next function.

```
int main()
{
  Distance d1;
  cout << "d1: ";
  d1.display(); cout << endl;
  Distance d2 = Distance(2, 5.9);
  Distance d3 = Distance(3.75);
  cout << "d2: "; d2.display(); cout << endl;
  cout << "d3: "; d3.display(); cout << endl;

  //test init helper function
  Distance d4 = Distance(5, 19.34);
  Distance d5 = Distance(100);
  cout << "d4: "; d4.display(); cout << endl;
  cout << "d5: "; d5.display(); cout << endl;

  //test add (<12 inches)
  cout << "d4 + d5: "; (d4 + d5).display(); cout << endl;
  //test add (>12 inches)
  cout << "d2 + d4: "; (d2 + d4).display(); cout << endl;

  //test sub (0 ft)
  cout << "d3 - d1: "; (d3 - d1).display(); cout << endl;
  //test sub (0 ft, negative conversion)
  cout << "d1 - d3: "; (d1 - d3).display(); cout << endl;

  //test sub (positive ft & inch)
  cout << "d4 - d2: "; (d4 - d2).display(); cout << endl;
  //test sub (negative ft & inch)
  cout << "d2 - d4: "; (d2 - d4).display(); cout << endl;

  //test sub (negative ft, positive inch)
  cout << "d4 - d5: "; (d4 - d5).display(); cout << endl;
  //test sub (positive ft, negative inch)
  cout << "d5 - d2: "; (d5 - d2).display(); cout << endl;


  //add more tests of your own ...


  return 0;
}
```

This main function should produce the following output:

```
d1: 0' 0"

d2: 2' 5.9"

d3: 0' 3.75"

d4: 6' 7.34"

d5: 8' 4"
```

d4 + d5: 14' 11.34"

d2 + d4: 9' 1.24"

d3 - d1: 0' 3.75"

d1 - d3: 0' 3.75"

d4 - d2: 4' 1.44"

d2 - d4: 4' 1.44"

d4 - d5: 1' 8.66"

d5 - d2: 5' 10.1"