Assignment 2: TimeSpan

Goals: Understanding operator overloading

Design and implement `TimeSpan` class which represents a duration in hours, minutes and seconds.

Displayed	**Meaning**
2:12:07	2 hours, 12 minutes, 7 seconds
-0:10:43	Minus 10 minutes and 43 seconds

The seconds and minutes have to be between 0 and 59. The hours can be 0 to any number. When `TimeSpan` is printed, the minutes and **seconds** (fixed 4/4) should always have 2 digits.

The constructor for `TimeSpan` can take 0, 1, 2, or 3 parameters. In addition to integer value, the constructor should also be able to handle double values and convert them as accurately as possible

TimeSpan(1.5, -10, 93)	same as 1:21:33	
TimeSpan(1.5)	same as 1:30:00	(**added 4/4 to clarify**)

Implement the following operators to work with `TimeSpan`:

1. Displaying: `operator<<`
2. Equality: `operator==, operator!=`
3. Comparison: `operator>, operator<, operator>=, operator<=`
4. Addition/Subtraction: `operator+, operator-, operator+=, operator-=`
5. Integer Multiplication

Integer multiplication allows us to multiply a `TimeSpan` with an integer and is not commutative. The other operators work on two given `TimeSpan` values and are commutative

```
TimeSpan ts3(1.5, 30.5, -90);
TimeSpan ts4;
ts4 = ts3 * 5;
// below line won't compile
// ts4 = 5 * ts3;
```

Sample assignment2.cpp, expand this as necessary to fully test your class:

```
#include <iostream>
#include <sstream>
#include <cassert>

#include "timespan.h"
```

```
using namespace std;

// testing constructor
void test1() {
   TimeSpan ts(1, 20, 30);
   stringstream ss;
   ss << ts;
   assert(ss.str() == "1:20:30");

   TimeSpan ts2(4, -20, -90);
   ss.str("");
   ss << ts2;
   assert(ss.str() == "3:38:30");

   TimeSpan ts3(1.5, 30.5, -90);
   ss.str("");
   ss << ts3;
   assert(ss.str() == "1:59:00");

   TimeSpan ts4(0, 0.07, 0);
   ss.str("");
   ss << ts4;
   assert(ss.str() == "0:00:04");
}

// testing equality, addition, subtraction, multiplication
void test2() {
   TimeSpan ts(1, 20, 30);
   TimeSpan ts2(1, 20, 30);
   TimeSpan ts3(0, 0, 0);
   assert(ts == ts2);
   assert(!(ts != ts2));
   assert(ts != ts3);
   assert((ts + ts + ts) == (ts2 * 3));
   assert((ts * 5) == (ts2 * 4) + ts2);
   assert((ts * 5) == (ts2 * 6) - ts2);
   assert((ts + ts - ts) == ((ts2 * 2) - ts));
   assert((ts - ts2) == ts3);
   assert((ts3 * 5) == ts3);
}

void testAll() {
   test1();
   test2();
}

int main() {
   testAll();
```

```
    cout << "Done." << std::endl;
    return 0;
}
```

Under unix, compile your code using

```
g++  -g -Wall -Wextra assignment2.cpp timespan.cpp -o assignment2
```

You need to submit `assignment2.zip` with the following files in it. See course assignments page for instructions on how to create it

`timespan.h` - the prototypes of `TimeSpan` class functions
`timespan.cpp` - the implementation of Library class functions
`assignment2.cpp` - tests demonstrating the Library constructor and functions
`output.txt` - See course assignments page for instructions on how to create it
`selfassessment.txt` - See course assignments page for the template

Tips & Hints

- If you have a constructor that can accept double parameters, you do not need a separate constructor that can accept integer parameters
- You should have a private `simplify()` function that can simplify any `TimeSpan`. This function should be called after any operation to make sure we still have valid `TimeSpan` representations.
- For `simplify`, first look at if the second is less than zero. If it is decrease minute by the appropriate amount to make second zero or a positive number. Next, check if second is greater than 59. If it is, decrease second by the appropriate amount and increase **minute** (fixed 4/4) accordingly. Do the same for minute, borrowing or adding to hour as necessary.