Object-Oriented Programming in C++

Assignment 1

This is the first of the laboratory exercises. The exercise consists of implementing a Time class for which the public interface has been specified as follows:

```
class Time

{

    public:
      // Constructors and destructor
      Time();           // Default constructor
      Time(Time const& time);         // Copy constructor
      Time(long secondsAfterMidnight);
      Time(char const* tstring);   // String in hh:mm:ss
                                   // format (24 hour time).
      ~Time();                     // Destructor
      // Const (read-only) functions
      char* GetTime(bool military = false) const;
      // Return string representation of the time.
      //          If military then hh:mm:ss
      //             else hh:mm:ss am.
      int   GetHour() const;        // Get hour value.
      int   GetMinute() const;      // Get minute value.
      int   GetSecond() const;      // Get second value.
      bool  operator !() const;     // Returns true if time
                                    //     is NOT valid
      bool  IsAM() const;           // Is time AM?
      bool  operator ==(Time const& time) const; // Are times equal?
      Time  operator +(Time const& time) const;  // Add times.
      Time  operator -(Time const& time) const;  // Subtract times.
      // Non-const (read/write) functions
      void  SetTime(int hrs, int mins = 0, int secs = 0);
      // Set the time to the values supplied (in 24 hour format)
      void  AddHours(int hours);          // Add hours (which may be <0).
      void  AddMinutes(int minutes);      // Add minutes (which may be < 0).
      void  AddSeconds(int seconds);      // Add seconds (which may be < 0).
};
```

Implement the specified functionality using whatever private data members and member functions you require.

This should be completed by Week 6 (11th August) and should include :

1. The completed TIME.CPP and TIME.H files. The TIME.CPP should be well commented.

2. You should test most, if not all, the logic paths in the class. The test harness should take the form of a main program, preferably driven by options selected by the tester.

Please bring progressive code to class, when it can be checked in tutorial time to ensure that you are on the right track.