CSCI 1300
Assignment 3: Time Keeper

Assigned: Wed Oct 8, 2003

Due: Tuesday Oct 21, 2003

Honor Code Reminder

Please adhere to the policy presented on the first day of class: All work you submit to be graded must be entirely your own. Breaking this rule will result in an F grade for the entire course. You may ask friends and family for help in understanding the syntax of the C++ language, for working through examples in the text, and for understanding compilation and run-time errors, but they may not write or help you write the code for any homework assignment. If you are stuck, see the professor, TAs, or undergraduate gurus.

The Assignment

The assignment is to do simple computations involving the time of day. The program will ask the user to specify a time of day, which we'll call the *start time*. This time is in the 12-hour hour:minute:second format, followed by "a" or "p" for a.m. or p.m. E.g., 1 minute after midnight is 12:01:00a; 1 minute and 1 second before midnight is 11:58:59p. We'll use the convention that midnight is 12:00:00a, and noon is 12:00:00p.

The program then asks for a *direction* to move in time (forward or backward) and a *duration* of time, and it prints the time of day when moving forward or backward by that duration. For example if the start time is 3:42:20p and the user asks for the time 1:05:10 (1 hour, 5 minutes, and 10 seconds) in the future, the program should output 4:47:30p. If the start time is 3:42:20p and the user asks for the time 0:50:00 in the past, the program should output 2:52:20p. The input and output will always be in a 12-hour hour:minute:second format, along with the a.m. or p.m. indicator.

Doing arithmetic (adding or subtracting time) with the 12-hour format is tricky. For this reason, there are two other ways of representing the time that you should find useful. One scheme is a 24-hour format, where midnight is 00:00:00, and the final second of the day is 23:59:59. The minutes and seconds are the same in the 12- and 24-hour formats, but the hour translates as follows:

12 hour format	24 hour format
12 a.m.	00
1–11 a.m.	01 – 11
12 p.m.	12
1 – 11 p.m.	13 – 23

Another format that is useful is *the number of seconds since the start of the day*; I'll call this the *seconds format*. There are 24x60x60 seconds in a day (hours/day x minutes/hour x seconds/minute). 12:00:00a (midnight) is second 0, 11:59:59p (the minute before midnight) is second 86399.

Sample Output

```
enter original time (hh mm ss a/p): 4 52 03 p
enter direction (+ or -): +
enter duration (hh mm ss): 3 10 00
The new time is 8:02:03p

enter original time (hh mm ss a/p): 8 30 30 a
enter direction (+ or -): -
enter duration (hh mm ss): 2 45 10
The new time is 5:45:20a

enter original time (hh mm ss a/p): 7 21 00 a
enter direction (+ or -): -
enter duration (hh mm ss): 8 10 00
The new time is 11:11:00p
```

```
enter original time (hh mm ss a/p): 4 20 00 a
enter direction (+ or -): +
enter duration (hh mm ss): 20 50 00
The new time is 1:10:00a

enter original time (hh mm ss a/p): 12 00 01 p
enter direction (+ or -): -
enter duration (hh mm ss): 0 0 02
The new time is 11:59:59a

enter original time (hh mm ss a/p): 11 59 59 p
enter direction (+ or -): +
enter duration (hh mm ss): 0 0 2
The new time is 12:00:01a
```

This is how your program should look when you run it. It should ask the user to enter (a) the original time, (b) the direction in time, "+" for forward and "–" for backward, and (c) the duration. Once the program prints the new time, it should begin again. You should make your program terminate when the hour of the original time is 0 (which is not a valid time). Note that the user should input the time separated by spaces (not colons), and should use "a" for a.m. and "p" for p.m. You do not have to do error checking for the input; you can assume the user will enter legal values. The output should be in a traditional 12-hour format.

Suggested Approach

There are many ways to solve this problem. If you want to tackle the problem with no further advice, stop reading here. The main goal of this assignment is to teach you how to write small, modular functions that lead to a compact, easy-to-read program, and that allow you to reuse code (i.e., call the functions from several places in your program). As you write each function, you should test it out by writing a short "main" function that calls the function you've written using arguments the user inputs and then prints out the return value from the function. If you test each individual function, then assembling them together should be straightforward.

In this section, I suggest a set of functions you could write to tackle the problem. Once you have written *and tested* these functions, you will find completing the assignment pretty straightforward.

`int convert_12hr_to_24hr(int hr, char ampm)`

This function accepts the hour in 12-hr format and outputs the hour in 24hr format. To do this conversion, the function needs to know both the hour (1-12) and whether it is a.m. or p.m. ('a' or 'p'). The function must perform the mapping summarized on the table on the previous page.

`int convert_24hr_to_seconds(int hr, int min, int sec)`

This function accepts the time in 24-hr format and outputs the time in the seconds format. The 24-hr time 00:00:00 should return 0 seconds; the 24-hr time 23:59:59 should return 86399 seconds.

`int sub_time (int hr1, int min1, int sec1, int hr2, int min2, int sec2)`

This function should take two times in 24-hr format and subtract the second from the first. It should do this by first converting the times to seconds format, and then simply performing a subtraction. It should return the time in seconds.

`int add_time(int hr1, int min1, int sec1, int hr2, int min2, int sec2)`

This function should take two times in 24-hr format and add the second to the first.

`int normalize_seconds(int sec)`

This function should take a time in seconds and *normalize* it so that the value lies between 0 and 86399. For example, a time of –1 is 1 second before the start of the day, and should be translated to 86399. A time of 863401 is 2 seconds past the end of the day, and should be translated to 1. This function will help you if the time shift duration crosses over from one day to another.

`void print_time(int sec)`

This function takes the time in seconds and prints out the time in 12-hr format. E.g., an input of 0 should print "12:00:00a". It will help you to first convert from seconds to the 24-hr format, and then from the 24-hr format to the 12-hr format. One hint: determining the count of hours from seconds is very much like what we did for the change program where you had to determine, say, the number of dollar bills to be returned from a change amount in pennies.

Handing in Your Program

Follow these instructions of your program will not be graded.

(1) By ***9:00 p.m.*** of the assignment due date *[NOTE THE CHANGE IN TIME]*, you must submit your program via the WebCT system. In a browser window, go to `webct.colorado.edu` and follow the directions to sign on. Click on "submit assignments" and "homework 3", and upload your C++ source code (the file with C++ instructions).

(2) The first line of your program should be a comment with the assignment #, your identikey ID (also your webct ID) and your name, e.g.,

```
    // Assignment 3    IDENTIKEY mozer      Michael C. Mozer
```

(3) The uploaded file *must* be given the same name as your identikey ID, e.g., `mozer.cxx`.

(4) If you have any sort of note to your TA (e.g., you weren't able to get some part of the program to work), write it in comments below the first line above. Your program should contain comments describing the operation of the program, and you should comment each block of code to explain at a high-level what the different parts of your program are doing. Comments should not simply paraphrase the C++ but give the reader and overview of what the code does.

(5) Late assignments will not be accepted.

(6) Come to recitation on the Wednesday following the due date with a copy of your program, either on floppy disk, or downloadable via email. You may be asked to demonstrate the program for your TA.

Grading

This assignment is worth a total of 100 points. Starting with this assignment, we will expect you to adhere to the class style guide, to which there is a link on the class home page. The style guide formalizes what I have presented in class. Each significant violation of the style guide will result in the loss of 5 points. Failing to produce output or accept input in the specified format will result in the loss of up to 10 points. Each significant bug will result in the loss of up-to 15 points. In particular, be concerned about *boundary conditions*, cases where the new time crosses over to a new day, or exactly hits a new day or the start of the day.

Unlike previous assignments, this one is not easily broken into parts, so you cannot expect to receive partial credit for doing part of the assignment. The purpose of this assignment is to teach you to write functions and assemble the functions into a working program. You will not achieve this goal by doing just part of the assignment. If you write all of the functions above and they have been tested and they work perfectly, but you don't assemble your program into a working whole, you will receive a maximum of 60 points.