# **HW\_3f** - **struct Time**

**-** Write a program that prompts the user for the hour, minutes, and seconds for a 24-hour time

(military time). If the time entered is valid, then one second is added to it, and the new

time is output to the screen

**The program should include:**

* **struct Time**
  + Declare a struct named: **Time**
    - The struct has 3 data members: **hours**, **minutes**, and **seconds** (all integers)
* In main(), declare an object of Time and name it: **time**
* Four functions in the program:

1. **getTime** – A void-returning function, gets input from user.
   * The object, time, is passed by reference to the function.

Note: Objects of structs and classes are always passed by reference.

* + The function prompts the user to enter the time.
  + The function reads the time entered by the user.
  + The function returns no value.

1. **isTimeValid** - Function returns a bool data type (true or false).
   * The function is called from the **getTime** function.
   * The parameter for this function is the time object that holds the hours, minutes,

and seconds representing the time entered by the user.

* + This function checks to make sure the time entered is valid.

**HINT:** if ((time.hours >=0) && (time.hours <= MAX\_HOURS)) && **…**

(if the time entered is invalid, return false, else return true)

* + - Include these global constants: MAX\_HOURS = 23;

MAX\_MINS = 59;

MAX\_SECS = 59;

* + If the function returns true, the getTime function is done and program execution

returns to main().

* + However, if the function returns false, then a message displays: Invalid time
  + Still within the *getTime* function, the user is again prompted for the time.
  + Again the time is read and passed to the i*sTimeValid* function, and so on.
  + Therefore, a do-while (or while) loop must be included within the *getTime* function.
    - Within the loop, the user is prompted for a time, and the *isTimeValid* function

is called.

Hint: do{

-----------

}while (validImput != true);

1. **addOneSecond** –A void-returning function adds one second to the time entered.

HINT: time.seconds++;

if (time.seconds > MAX\_SECS)

{

time.seconds = 0;

time.minutes++;

if (……………..

1. **displayTime** – A void-returning function displays the time after the second has

been added.

**cout.fill(‘0’)** - This library function will fill any empty spaces with the specified

character. It checks to see what setw is, and then fills if needed.

Ex: cout << setw(2); // Remember that setw() must be placed in front

// of every number that is to be displayed.

cout.fill(‘0’);

cout << time.hours;

* Include a **do/while loop**, along with **system(“cls”)** – (Allows user to repeat program)

/\* **OUTPUT #1**

Enter the time in "military time", (24-hour format),

in the following order: HH:MM:SS, (Hours, Minutes, Seconds).

Hours: 14

Minutes: 44

Seconds: 22

After adding one second, the time is 14:44:23.

Do it again? (Y/N) y

/\* **OUTPUT #2**

Enter the time in "military time", (24-hour format),

in the following order: HH:MM:SS, (Hours, Minutes, Seconds).

Hours: 1

Minutes: 9

Seconds: 59

After adding one second, the time is 01:10:00.

Do it again? (Y/N) y

/\* **OUTPUT #3**

Enter the time in "military time", (24-hour format),

in the following order: HH:MM:SS, (Hours, Minutes, Seconds).

Hours: 21

Minutes: 59

Seconds: 59

After adding one second, the time is 22:00:00.

Do it again? (Y/N) y

/\* **OUTPUT #4**

Enter the time in "military time", (24-hour format),

in the following order: HH:MM:SS, (Hours, Minutes, Seconds).

Hours: 23

Minutes: 59

Seconds: 59

After adding one second, the time is 00:00:00.

Do it again? (Y/N) y

/\* **OUTPUT #5**

Enter the time in "military time", (24-hour format),

in the following order: HH:MM:SS, (Hours, Minutes, Seconds).

Hours: 27

Minutes: 23

Seconds: 11

Invalid data.

Enter the time in "military time", (24-hour format),

in the following order: HH:MM:SS, (Hours, Minutes, Seconds).

------ \*/