# Simple Selections

Workshop 5

In this workshop, you will code and execute a C-language program that accepts times from the user, input as hours, minutes and seconds. The program will add the times and produce the output using both a 12 and 24 hour clock.

## LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities:

- to declare variables of integral types
- to code a simple calculation using C operators and expressions
- to accept numerical values from the user using scanf
- to use modulus and integer division to perform calculations
- to use if statements to implement logic

## INSTRUCTIONS

---

*Part I (IN-LAB 30%)*

---

Write a program, `add_hms24.c`, that starts by asking the user to enter a time as hours, minutes and seconds using a 24 hour clock. The program will print the following message:

```
Enter an initial 24-hour time as hours minutes and seconds (0 to stop):
```

The program will then ask the user to enter an amount in hours, minutes and seconds to add:

```
Enter the time to add as hours minutes and seconds:
```

The program will add these two times together and print the time that results from adding the indicated number of hours, minutes and seconds to the initial time of day. The result will be printed using a 24 hour clock. Your program must not convert the time to fractional days but should add them as separate hours, minutes and seconds.

Sometimes adding times will cause a carry over, which you will need to take into account in your calculations. For example, adding 7:30:50 + 1:29:50 = 9:00:40. When we add the seconds, they yield 100, which is 1 minute and 40 seconds, so we need to carry the extra minute over to the minute column. Since the minutes already sum to 59, this makes the minutes go to 60. But 60 minutes is the next hour, so we make the minutes zero and carry a over to the hour. You should, and MUST, use modulus and integer division in your calculations.

The number of days can be calculated by dividing the hours resulting from the calculation by 24 and, if it is greater than 0, you know it is not today. If the resulting time is tomorrow, your program should print the time followed by the word “tomorrow”. If it is more than 1 day in the future, it should print something like “in 3 days” after the time.

### Execution and Output Example (Multiple runs of the program):

```
Enter an initial 24-hour time as hours minutes and seconds: 7 30 0
Enter the time to add as hours minutes and seconds: 5 0 0
Total is 12:30:00

Enter an initial 24-hour time as hours minutes and seconds: 7 30 0
Enter the time to add as hours minutes and seconds: 8 30 0
Total is 16:00:00

Enter an initial 24-hour time as hours minutes and seconds: 7 30 0
Enter the time to add as hours minutes and seconds: 32 30 0
Total is 16:00:00 tomorrow

Enter an initial 24-hour time as hours minutes and seconds: 7 30 0
Enter the time to add as hours minutes and seconds: 80 30 0
Total is 16:00:00 in 3 days

Enter an initial 24-hour time as hours minutes and seconds: 7 30 50
Enter the time to add as hours minutes and seconds: 1 58 50
Total is 9:29:40
```

## SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above. Failure to use the submitter will incur a penalty of 40%.

If not on matrix already, upload your **add_hms12.c** to your matrix account. Compile and run your code and make sure everything works properly.

```
AtThePrompt> gcc -Wall add_hms12.c –o ws<ENTER>
```

-Wall activates the display of warnings in GCC compiler.
-o sets the executable name (ws)

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

**~profname.proflastname/submit btm200/w5_lab <ENTER>**

and follow the instructions.

---

*Part II (AT-HOME 40%)*

---

Create a new program, add_hms12.c, that will extend your existing program to work with a 12 hour clock rather than a 24 hour clock. The program will accept the initial times using a 24 hour clock, as before, but will print out the results using a 12 hour clock that appends “AM” or “PM” to the times.

### Execution and Output Example (Multiple runs of the program):

```
Enter an initial 24-hour time as hours minutes and seconds: 7 30 50
Enter the time to add as hours minutes and seconds: 1 29 50
Total is 9:00:40 AM

Enter an initial 24-hour time as hours minutes and seconds 7 30 0
Enter the time to add as hours minutes and seconds: 5 0 0
Total is 12:30:00 PM

Enter an initial 24-hour time as hours minutes and seconds: 7 30 0
Enter the time to add as hours minutes and seconds: 25 30 0
Total is 9:00:00 AM tomorrow

Enter an initial 24-hour time as hours minutes and seconds 7 30 0
Enter the time to add as hours minutes and seconds: 49 30 0
Total is 9:00:00 AM in 2 days
```

## AT-HOME REFLECTION (30%)

In a text file named **reflect.txt**, answer the following:

- Explain how modulus is used in your calculations. Describe one other technique for performing the calculations and compare it to the use of modulus. Does modulus simplify the calculations?
- Adding times is similar to adding two 3 digit numbers. You add them digit by digit and, when the sum exceeds 9, you do a carry. Explain how adding number is similar to adding times and where the two techniques differ and why.
- Assume that, instead of adding hours, minutes and seconds, you need to add years, months and days. What new problems would you encounter solving this problem that you did not have to worry about in the addition of times? Describe how you would solve these problems.

## SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **add_hms12.c** and **reflext.txt** to your matrix account. Compile and run your code and make sure everything works properly.

```
AtThePrompt> gcc -Wall add_hms12.c –o ws<ENTER>
```

-Wall activates the display of warnings in GCC compiler.
-o sets the executable name (ws)

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit btm200/w5_home <ENTER>
```

and follow the instructions.

## SUBMISSION POLICY

The workshop is **due within 4 days after your lab by 23:59.**
**All your work (all the files you create or modify) must contain your name, Seneca email and student number. Failure to use the submitter will incur a penalty of 40%.**
You are responsible for regularly backing up your work.

**Please Note**

- A successful submission does not guarantee full credit for this workshop.
- If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.