Solution: Fix the Code

This lesson provides a solution to the challenge given in the previous lesson.

Solution

Here is the program that will produce the desired output on entering 06:09 20:0 as input.

```
import std.stdio;
import std.string;
import std.exception;

/* Reads the time as hour and minute after printing a
 * message. */
void readTime(string message,
              out int hour,
              out int minute) {
    write(message, "? (HH:MM) ");

    readf(" %s:%s", &hour, &minute);

    enforce((hour >= 0) && (hour <= 23) &&
            (minute >= 0) && (minute <= 59),
            "Invalid time!");
}

/* Returns the time in string format. */
string timeToString(int hour, int minute) {
    assert((hour >= 0) && (hour <= 23));
    assert((minute >= 0) && (minute <= 59));

    return format("%02s:%02s", hour, minute);
}

/* Adds duration to start time and returns the result as the
 * third pair of parameters. */
void addDuration(int startHour, int startMinute,
                 int durationHour, int durationMinute,
                 out int resultHour, out int resultMinute) {
```

```
    resultHour = startHour + durationHour;
    resultMinute = startMinute + durationMinute;

    resultHour += resultMinute / 60; resultHour %= 24;
    resultMinute %= 60;
    assert((resultHour >= 0) && (resultHour <= 23));
    assert((resultMinute >= 0) && (resultMinute <= 59));
}

void main() {
    int startHour;
    int startMinute;
    readTime("Start time", startHour, startMinute);

    int durationHour;
    int durationMinute;
    readTime("Duration", durationHour, durationMinute);

    int endHour;
    int endMinute;
    addDuration(startHour, startMinute,
                durationHour, durationMinute,
                endHour, endMinute);

    writefln("%s hours and %s minutes after %s is %s.",
             durationHour, durationMinute,
             timeToString(startHour, startMinute),
             timeToString(endHour, endMinute));
}
```

Program to calculate the end time

Solution explanation

The problem is the following `assert` check:

```
assert((hour >= 0) && (hour <= 23));
```

The reason is that `addDuration()` can produce hour values that are greater than 23. Adding a modulo operation on `resultHour` at the end of this function will partially guarantee correct output.:

```
void addDuration(int startHour, int startMinute,
                 int durationHour, int durationMinute,
                 out int resultHour, out int resultMinute) {
    resultHour = startHour + durationHour;
    resultMinute = startMinute + durationMinute;

    if (resultMinute > 59) {
        ++resultHour;
    }
    }

    resultHour %= 24;


}
```

addDuration function

Observe that the function has other problems. For example, `resultMinute` may end up being greater than 59. The following function calculates the minute value correctly and makes sure that the function's output guarantees are enforced:

```
void addDuration(int startHour, int startMinute,
                 int durationHour, int durationMinute,
                 out int resultHour, out int resultMinute) {
    resultHour = startHour + durationHour;
    resultMinute = startMinute + durationMinute;

    resultHour += resultMinute / 60; resultHour %= 24;
    resultMinute %= 60;
    assert((resultHour >= 0) && (resultHour <= 23));
    assert((resultMinute >= 0) && (resultMinute <= 59));
}
```

addDuration function

In the next lesson, you will find a quiz to test your understanding of the concepts covered in this chapter.