Programming for computerteknologi
Hand-in Assignment Exercises

Week 12: Bundling Data and their Functions Together

Exercise 1)

In this exercise we have been given the task to program a class in C++ called *Duration*.
I have started by creating both a header file and a cpp file and created a class called Duration. It looks like the following.

```cpp
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

class Duration
{
private:
    int time;
    int alarm;
    bool alarmHasBeenSet;
    bool checkAndUpdateAlarm();
public:
    Duration();
    Duration(int t);
    ~Duration();

    int getDuration();
    int getAlarm();
    bool checkAlarm();

    bool tick();
    bool tick(int t);

    void setAlarm(int t);
};
```

I have made both a constructor with no given time and a constructor with a given time, which sets the private attributes to either a given time or create the different attributes by setting them to 0.

```
3   /* Constructor */
4   Duration::Duration() {
5       time = 0;
6       alarm = 0;
7       alarmHasBeenSet = false;
8   }
9
10  /* Contructor with given time */
11  Duration::Duration(int t) {
12      assert(t >= 0);
13      alarm = 0;
14      alarmHasBeenSet = false;
15      time = t;
16  }
```

Next up I have created a way to add time to the private attribute timer.

```
33  /* Add 1 to time */
34  bool Duration::tick() {
35      time++;
36      return checkAndUpdateAlarm();
37  }
38
39  /* Add given number to time */
40  bool Duration::tick(int t) {
41      assert(t > 0);
42      time += t;
43      return checkAndUpdateAlarm();
44  }
```

The functions is boolean functions that returns either true of false if the timer exceeds the alarm clock.

```
53  /* Checks and resets alarm if necessary */
54  bool Duration::checkAndUpdateAlarm() {
55      if(alarmHasBeenSet == false) {
56          return false;
57      } else {
58          if(time >= alarm) {
59              alarm = 0;
60              alarmHasBeenSet = false;
61              return true;
62          } else {
63              return false;
64          }
65      }
66  }
```

The alarm is defined by the following function:

```
46  /* Set alarm */
47  void Duration::setAlarm(int t) {
48      assert(t > time);
49      alarm = t;
50      alarmHasBeenSet = true;
51  }
```

To test my functions, I have made a separate “test.cpp” file. I have also made sure to include different asserts in the functions shown on the illustrations above to assure the program will run correctly.

```
12  TEST_CASE("duration") {
13      /* test exercise b) - Duration() */
14      Duration *d0 = new Duration();
15      REQUIRE(d0->getDuration() == 0);
16
17      /* test exercise c) - Duration(int t) */
18      Duration *d1 = new Duration(7);
19      REQUIRE(d1->getDuration() == 7);
20
21      /* test exercise e) - tick() */
22      REQUIRE(d0->getDuration() == 0);
23      d0->tick();
24      REQUIRE(d0->getDuration() == 1);
25      d0->tick();
26      REQUIRE(d0->getDuration() == 2);
27
28      /* test exercise f) - tick(int t) */
29      Duration *d2 = new Duration();
30      REQUIRE(d2->getDuration() == 0);
31      d2->tick(47);
32      REQUIRE(d2->getDuration() == 47);
33      d2->tick(65);
34      REQUIRE(d2->getDuration() == 112);
```

```
36      /* test exercise g) - Alarm */
37      Duration *d3 = new Duration();
38      REQUIRE(d3->getDuration() == 0);
39      REQUIRE(d3->getAlarm() == 0);
40      REQUIRE(d3->checkAlarm() == false);
41      d3->setAlarm(5);
42      REQUIRE(d3->getAlarm() == 5);
43      REQUIRE(d3->checkAlarm() == true);
44      REQUIRE(d3->tick(4) == false);
45
46      REQUIRE(d3->getAlarm() == 5);
47      REQUIRE(d3->checkAlarm() == true);
48
49      REQUIRE(d3->tick() == true);
50      REQUIRE(d3->getAlarm() == 0);
51      REQUIRE(d3->checkAlarm() == false);
```

```
===============================================================================
All tests passed (19 assertions in 1 test case)
```