Programmeren (FEB22012)

7. Exercise

Deadline for submission: 2011-09-25 23:59 CET

Instructions

Testing your code in an automated manner is very important for development of working programs. Most programs are not written for one-time use, but they develop iteratively, and new features are added with time. However, adding new features can break existing functionality. *Unit tests* can help to give you confidence in your program code by testing functionality at the granularity of a single method call. For example, if you had a function that computes the input + 2:

```
function a = addTwo(b)
        a = b+2;
end
```

you could make a unit test for testing the function with some input values:

```
function testAddTwo()
        assert(addTwo(2) == 4);
        assert(addTwo(-2) == 0);
        "addTwo OK"
        end
end
```

Now although you cannot be sure about whether the addTwo works properly with all possible inputs (as you never can be sure, remember the halting problem?), the unit test gives you some assurance that the addTwo-method works.

Exercise

- Write a function that takes as parameter a string representing a number in binary format (e.g. 1001001) and returns its base 10 representation (e.g. 73). Do the computation with iteration and comment loop invariant above the loop.
- Write a function that takes as a parameter a floating point number and returns the components of its IEEE 754 representation (sign, fraction, and exponent).
- Implement unit tests for these functions.