Prof. Dr. Peter Thiemann
Manuel Geffken
Matthias Keil Summer Term 2013

Softwaretechnik

http://proglang.informatik.uni-freiburg.de/teaching/swt/2013/

Exercise Sheet 7

Exercise 1

Given the following program with input variables `x`, `y` and expressions `e1` . . . `e8`. The expressions `e1` . . . `e8` will not change the program variables.

```
e1;
while(x > 0) {
   e2;
   if(y > 0) {
      e3;
   } else {
      e4;
   }
   if(x mod y == 0) {
      e5;
   } else {
      e6;
   }
   e7;
}
e8;
```

1. Create a set of *Test Cases* to achieve *Full Line Coverage*. How many *Test Cases* do you need?
2. Create a set of *Test Cases* to achieve *Full Branch Coverage*. How many *Test Cases* do you need?
3. Create a set of *Test Cases* to achieve *Path Coverage*. How many *Test Cases* do you need?
4. Assume that the expression `e2` will increase the value of `y`. How many *Test Cases* do you need to achieve a *Full Branch Coverage*? Justify your answer.
5. Assume that the expression `e7` will decrease the value of `x`. How many *Test Cases* do you need to achieve a *Full Path Coverage*? Justify your answer.
6. Which kind of software testing is this?

Exercise 2

Given the following function `sort` with input `X[] x`. The function specification requires a list with elements of type `X`.

```
public X[] sort(X[] x) {...}
```

Provide *Test Cases* for *Black-Box-Testing* to specify if the program works correctly. How many test cases do you need? Justify your answer and describe the purpose behind each testcase.

Exercise 3

Regard the *Test Quiz* shown in the lecture. You will have a simple program reading four integers from the command line. Each value represents the length of one side of a quadrangle *(A-B-C-D)*. The program will tell you whether the input describes a valid quadrangle or not and will divide the quadrangle in one of the following groups.

square four equal sides

rectangle two pairs of equal opposite sides

kite two pairs of equal-length sides

quadrangle

invalid quadrangle

Create a set of *Test Cases* to verify the functionality of this program. Treat special cases and permutations of the input as well as overlappings.

Exercise 4

In the previous exercises, we have examined the specification of programs using pre- and postconditions. In this exercise, we consider the useof examples for explaining the behavior of a program. To this end we will use Pex, a tool from Microsoft Research that creates a set of test cases by analysing the source code. We will see that it is usually harder to understand the semantics of a program if a set of test cases is given instead of a specification.
Familiarize yourself with Pex4Fun at `http://www.pexforfun.com/`. Provide code that matches a secret implementation. Test your solution by asking Pex. Pex either returns true if your solution is correct, or provides a counter-example for parameters for which your solution fails.

1. Provide code that matches the implementation of *Puzzle* at `http://goo.gl/t5SPC`. What does *Puzzle* compute? *Hint:* Consider the triangle example discussed in the lecture.

2. Provide code that matches the implementation of *Puzzle* at `http://goo.gl/SZVZS`. What does *Puzzle* compute?