

## Software Quality Assurance (WS12/13)

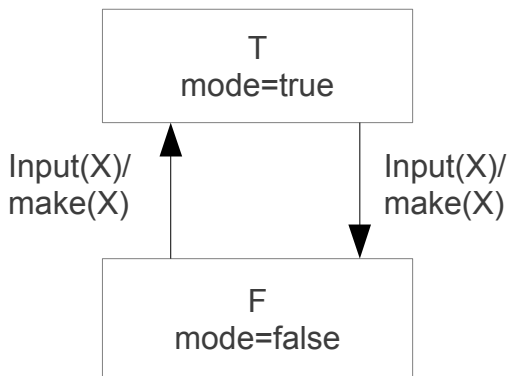
### Problem Set 6

Due: in exercise, 30.01.2013

#### Problem 1: Data Flow Anomaly Analysis

Consider the following Java implementation of the class Switch. The class is a module of a technical application which is classified as safety-critical.

A switch provides a return value which depends on the internal state and the given input value. The return value is true if input and internal state coincide, else it is false. Every usage modifies the internal state of the switch.



```
01 public class Switch {
02     private boolean mode;
03     public Switch() {
04         init();
05     }
06     private void init() {
07         mode=true;
08     }
09     public boolean toggle(boolean mode) {
10         boolean result;
11         if((this.mode&&mode)||(!this.mode&&!mode))
12             result=true;
13         else
14             result=false;
15         if (this.mode)
16             this.mode=!mode;
17         else
18             mode=mode;
19         return result;
20     }
21 }
```

- Judge whether the class Switch complies with the given requirements.
- Perform a data flow anomaly analysis for the operation `toggle`.

## Problem 2: Data Flow Anomaly Analysis

A software company develops software packages for commercial animal housing. A particular function, which is implemented in the C programming language, computes the daily amount of feed for different animal species depending on their individual weight.

Until now, this function was only part of a software package for farms and worked failure-free since years. Recently, it is also included in a software package for zoological gardens and it produces wrong output in some cases. By performing a data flow analysis, the faults should be revealed.

```
/* Own data type for enumeration of animal species */
typedef enum {COW, HORSE, PIG, ELEPHANT} Animal_A;

/* Function for determining the daily amount of feed depending
 * on the animal species and the individual weight
 */
01 float feedamount(Animal_A species, float weight)
02 {
03     float amount, factor;
04     switch (species)
05     {
06         case COW:
07         {
08             factor = 0.05;
09             break;
10         }
11         case HORSE:
12         {
13             factor = 0.1;
14             break;
15         }
16         case PIG:
17         {
18             factor = 0.02;
19             break;
20         }
21     } // end switch
22     amount = factor * weight;
23     return amount;
24 } // end feedamount
```

- a) What mistakes were performed and how would the consequences have been avoided?
- b) Perform a data flow anomaly analysis for the operation feedamount.

### Problem 3: Data Flow Anomaly Analysis

Consider the following Java implementation of the operation ALL\_POSITIVE which checks whether all elements of a one-dimensional array are positive. As parameters, the field and its length are given.

```
01 boolean ALL_POSITIVE(int[] array,int len) {
02     boolean result;
03     int i,tmp;
04     i=0;
05     result=true;
06     while (i<len&&result) {
07         tmp=array[i];
08         if (tmp<=0)
09             result=false;
10         i++;
11     }
12     return result;
13 }
```

Perform a data flow anomaly analysis for the operation ALL\_POSITIVE.

### Problem 4: Slicing

Create static backward slices for Variables result, mode and this.mode.

```
09 public boolean toggle(boolean mode) {
10     boolean result;
11     if((this.mode&&mode)||(!this.mode&&!mode))
12         result=true;
13     else
14         result=false;
15     if (this.mode)
16         this.mode=!mode;
17     else
18         mode=mode;
19     return result;
20 }
```