Vrije Universiteit, Department of Computer Science

Examination paper for **Software Testing**
23 May 2016 12:00-14:45

This is a closed book written exam.

No printed material or electronic devices are admitted for use during the exam.

The answers have to be given in English.

Both homework and exam are compulsory and graded on a 1 to 10 scale.

The exam grade is calculated as (Q1+Q2+Q3+Q4+Q5 +10)/10.

The final grade is calculated as 0.6*homework + 0.4*exam

A pass is given only if both homework and exam components are >= 5.5

	Q1 (concepts)	**Q2**	**Q3**	**Q4**	**Q5 (code)**	**Σ Qi**	**Maximum credits= (ΣQi+10)/10**
a)	**3**				**3**		
b)	**6**				**5**		
c)	**6**				**3**		
d)	**8**				**5**		
e)	**6**				**3**		
f)	**6**				**3**		
g)							
Total	**35**	**12**	**9**	**12**	**22**	**90**	**10**

Good luck!

Q1. Concepts [35p]

a. What is the main goal of testing safety critical systems? [3p]
b. Enumerate and explain shortly the properties of a good software requirements specification. [6p]
c. Define verification and validation, and draw the V-model for software testing where you show where activities happen. [6p]
d. The Oosterscheldekering (in English: Eastern Scheldt storm surge barrier), is the largest of the Delta Works series of dams and storm surge barriers, designed to protect the Netherlands from flooding from the North Sea. The principle is that when the water level reaches at least 3m above sea level, the system shuts the doors to the sea. Apply the first step of STAMP analysis to this storm surge barrier. Identify a possible accident and possible hazards. Draw the control structure and identify the possible unsafe control actions. [8p]
e. Explain what symbolic execution is and give an example, showing how it can be used to generate test inputs. [6p]
f. Define metamorphic testing and demonstrate it on the following code: [6p]

```
public static int addValues(int a[]){
      int sum=0;
      for(int i=0;i<a.length;i++){
            sum+=a[i];}
      return sum;}
```

Testing from requirements Q2-Q4 [33]

Q2. [12p]

Consider the following requirement:

[REQ 631] The system shall allow shipments for which the price is less than or equal to €100.

Design and generate test cases to defensively test this requirement by using equivalence partitioning (EP) and boundary value analysis (BVA). Justify your test cases specifications and minimize your test cases.

Q3. [9p]

Consider the following requirements.

The system shall only calculate discounts for members
The system shall calculate a discount of 5% if the value of the purchase is less than or equal than 100 euros. Otherwise the discount is 10%.
The system should write the discount percentage on the invoice.
The system must write in the invoices of nonmembers that memberships give discount.

Test the calculation and printing on the invoice features using a decision table approach.

Q4. [12p]

This is an admission requirement the Boise State's University:

A high school graduate is admitted based on the high-school grade point average (GPA) and the ACT composite score (English, mathematics, reading, science).
The GPA score is a number, 2.0<=GPA<=4.0 (for example 2.8) and ACT is a number, 16 <= ACT <= 36, (for example 30).
Admission is possible only if 10*GPA + ACT >= 56.

Design test cases for this requirement using 1x1 domain testing.

a) Draw the valid domain. [3p]
b) Construct the domain test matrix. [9p]

Q5. Code based testing [22p]

Given the following program:

```
public static double CalculateZ (double x, double y) {

     double z = 2;
     if (y <= 0) {
               z = y*x + 1 ;
     }
     z = z * x + 1 ;
     return (z) ;
}
```

a) Draw the control flow graph. [3p]
b) Generate test cases for reaching complete statement coverage. Use as few test cases as possible. [5p]
c) Generate test cases for reaching complete decision coverage. Use as few test cases as possible. [3p]
d) Generate a test suite that is adequate with respect to all –uses criterion. Use as few test cases as possible. [5p]
e) Generate a mutant and design a test case that will kill it. [3p]
f) Generate an equivalent mutant. [3p]