Testing and Verification (DIT085)
Final Examination - April 13, 2015

Important Notes. It is not allowed to use study material, computers, and calculators during the examination. The examination comprises 4 question in 2 pages. Please check beforehand whether your copy is properly printed. In order to obtain a VG you need to obtain 80/100, for a G you need to obtain 60/100. Give complete explanation and do not confine yourself to giving the final answer. The answers may be given in Swedish or English. The solutions to the exercises will be available after the examination through the course page. **Good luck!**

Exercise 1 (20 points) Define the following concepts:

1. gaps and redundancies,
2. finite feasibility,
3. nearest inverse denominator, and
4. fault, error, and failure.

Solution.

1. Gaps are those parts of the implementation that are not tested by a test-suite. Redundancies are those parts that are repeatedly tested. Gaps and redundancies are typical problems of functional testing.
2. A coverage criterion is finitely feasible if for each implementation, there is a finite test suite satisfying it.
3. A node in the control flow graph is an inverse denominator of the other, if the former appears in all paths from the latter to termination. A node is the nearest inverse denominator of another node, if it is an inverse denominator of the latter and it appears before the other inverse denominators in some simple path.
4. A fault is a mistake made in (the absence of) some implementation. The infected state, which is the result of executing a faulty implementation is an error. A failure is the observable result of an error, e.g., in the program output.

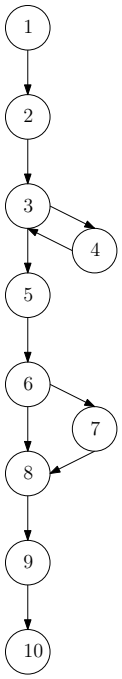
Exercise 2 (50 points) Consider the following program.

```
1: Input(x);
2: Input(y);
3: while x < y then
4:     x := x + 1;
5: end while
6: if y <= 20 then
7:     y := y + x;
8: end if
9: x := 2 * y
10: Output(x);
```

1. Draw the control-flow graph of the program (5 pts),
2. Calculate all prime paths of the control-flow graph. (10 pts),
3. Define a test suite that satisfies all DU-paths testing; for each test-case give the DU-path covered by it (10 pts), and
4. Calculate $Slice(10, \{x\})$. The final solution is not sufficient; you need to elaborate on the steps towards the final solution (include the relevant variables and the approximations towards the final slice). (25 pts)

Solution.

1. The CFG of the program is given below:



2. Prime paths (simple paths that are not included in any other simple path) are given below:

   [1, 2, 3, 5, 6, 8, 9, 10]

   [1, 2, 3, 5, 6, 7, 8, 9, 10]

   [3, 4, 3]

   [4, 3, 4]

3. The set of DU-paths (including the immediate consequence of conditions) are given below:

   For x:

   [1, 2, 3, 4]

   [1, 2, 3, 5]

   [1, 2, 3, 5, 6, 7]

   [4, 5, 6, 7]

   [9, 10]

   For y:

[2, 3, 4]

[2, 3, 5]

[2, 3, 5, 6, 7]

[2, 3, 5, 6, 8]

[2, 3, 5, 6, 8, 9, 10]

[7, 8, 9, 10]

The following set of test cases satisfies the all DU-paths criterion:

inputs: x = 1, y = 2, expected output x = 8, covers: [1, 2, 3, 4] [2, 3, 4] [4, 5, 6, 7] [9, 10]

inputs: x = 3, y = 2, expected output x = 10, covers: [1, 2, 3, 5] [1, 2, 3, 5, 6, 7] [2, 3, 5] [2, 3, 5, 6, 7] [7, 8, 9, 10]

inputs: x = 22, y = 21, expected output x = 42, covers: [2, 3, 5, 6, 8] [2, 3, 5, 6, 8, 9, 10]

4. The slice is calculated as follows:

m	**DEF(m)**	**Relevant$_0$(m)**	**Slice$_0$**	**Cond$_1$**	**Rel$_1$**	**Slice$_1$**	**Slice**
1	$\{x\}$	$\emptyset$	$\surd$	$\times$	$\emptyset$	$\surd$	$\surd$
2	$\{y\}$	$\{x\}$	$\surd$	$\times$	$\{x\}$	$\surd$	$\surd$
3	$\emptyset$	$\{y, x\}$	$\times$	$\surd$	$\{y, x\}$	$\surd$	$\surd$
4	$\{x\}$	$\{y, x\}$	$\surd$	$\times$	$\{y, x\}$	$\surd$	$\surd$
5	$\emptyset$	$\{y, x\}$	$\times$	$\times$	$\{y, x\}$	$\times$	$\surd$
6	$\emptyset$	$\{y, x\}$	$\times$	$\surd$	$\{y, x\}$	$\surd$	$\surd$
7	$\{y\}$	$\{y, x\}$	$\surd$	$\times$	$\{y, x\}$	$\surd$	$\surd$
8	$\emptyset$	$\{y\}$	$\times$	$\times$	$\{y\}$	$\times$	$\surd$
9	$\{x\}$	$\{y\}$	$\surd$	$\times$	$\{y\}$	$\surd$	$\surd$
10	$\emptyset$	$\{x\}$	$\times$	$\times$	$\{x\}$	$\times$	$\times$
		$\{x\}$			$\{x\}$		

Exercise 3 (20 points) Consider the following properties in the TCTL language. Describe them in words (English/Swedish):

1. `A⟨⟩ deadlock` (5 pts),
2. `A[]` (`m.s and (not mp.sp)`) (5 pts),
3. `m.s` $\longrightarrow$ `mp.sp` (10 pts),

Solution.

1. There is some execution in which some state does not have any outgoing enabled transition,
2. For all executions eventually a state is reached in which process (automaton instance `a` is either in location `l` or in location `lp` and the value of global variable `v` is at most 2,
3. For all execution if process `a` is in location `l`, then eventually in all future executions `a` will be in location `lp`.

Exercise 4 (10 points) Regarding Visual GUI testing, explain the benefits of VGT (versus GUI model testing and manual GUI testing).

Solution. (1) Speed and test frequency, (2) Flexibility, (3) Higher defect finding ability, (4) High usability and easy to learn and (5) feasible maintenance costs.