6.0 ECTS/4.5h VU Programm- und Systemverifikation (184.741) June 15, 2021				
Kennzahl (study id)	Matrikelnummer (student id)	Familienname (family name)	Vorname (first name)	Platz (seat)

1.) Coverage

Consider the following program fragment and test suite:

```
unsigned gcd (unsigned x, unsigned y)
{
  unsigned min, max, t;
  if (x<y) {
    min = x;
    max = y;
  } else {
    min = y;
    max = x;
  }
  for (t = min; t>0; t--) {
    if ( (x%t==0) && (y%t==0) )
        return t;
  }
  return max;
}
```

Inputs		Outputs
x	y	return value
0	1	1
1	0	1
2	3	1

Remarks:

- Here, branch coverage also requires unconditional branches (return) to be covered!
- `t--` is short for `t = t - 1`.
- Variable declarations are not definitions.

(a) **Control-Flow-Based Coverage Criteria**

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above.

	satisfied	
Criterion	yes	no
statement coverage		
decision coverage		
branch coverage		
MC/DC		

For each coverage criterion that is *not* satisfied, explain why this is the case:

(5 points)

(b) **Data-Flow-Based Coverage Criteria**

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (here, the parameters of the function do not constitute definitions, the `return` statement is a c-use):

	satisfied	
Criterion	yes	no
all-defs		
all-p-uses		
all-c-uses		
all-c-uses/some-p-uses		
all-du-paths		

For each coverage criterion that is not satisfied, explain why this is the case:

(8 points)

(c) Consider the two coverage criteria below.

- If the test-suite from above does not satisfy the coverage criterion, augment it with the *minimal* number of test-cases such that this criterion is satisfied. If full coverage cannot be achieved, explain why.
- If the coverage criterion is already achieved, explain why.

all-c-uses

Inputs		Outputs
x	y	result

MC/DC

Inputs		Outputs
x	y	result

(2 points)

2.) Hoare Logic

Prove the Hoare Triple below (assume that t variable `m` is of type unsigned integer). You need to find a sufficiently strong loop invariant.

Annotate the following code directly with the required assertions. Justify each assertion by stating which Hoare rule you used to derive it, and the premise(s) of that rule. If you strengthen or weaken conditions, explain your reasoning.

Note: No points for assertions that were not clearly derived by using one of the rules from the lecture!

```
{true}

①

if ((m + n) % 2 != 0) {

  ②

  m = m + 1;

  ③

} else {

  ④

  skip;

  ⑤

}

⑥

while ((m != 0) && (n != 0)) {

  ⑦

  m = m - 1;

  ⑧

  n = n - 1;

⑨

}

⑩

{(m%2 == 0)}
```

(10 points)

3.) Invariants Consider the following program, where `a`, `b`, `x` and `y` are integer values in $\mathbb{Z}$ (that means no over- or underflow can happen):

```
if (x == y) {
  a = b;
}

while (x < 42) {
  x = x + 1;
  y = y + 1;
}
```

Consider the formulas below; tick the correct box (☑) to indicate whether they are loop invariants for the program above.

- If the formula is an inductive invariant for the loop, provide an informal argument that the invariant is inductive.
- If the formula P is an invariant that is *not* inductive, give values of `x` and `y` before and after the loop body demonstrating that the Hoare triple

$$\{P \wedge B\} \quad \mathtt{x} = \mathtt{x} + 1;\ \mathtt{y} = \mathtt{y} + 1; \quad \{P\}$$

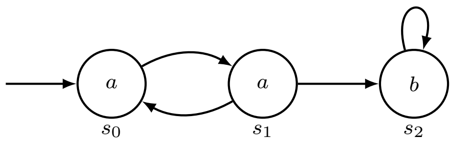
  (where B is `(x < 42)`) does not hold.
- Otherwise, provide values of `a`, `b`, `x`, and `y` that correspond to a reachable state showing that the formula is *not* an invariant.

$(a - b) = (x - y)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(a \neq b) \vee (x = y)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(x \neq y) \vee (a = b)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			

(10 points)

4.) Temporal Logic

(a) Consider the following Kripke Structure:



For each formula, give the states of the Kripke structure for which the formula holds. In other words, for each of the states from the set $\{s_0, s_1, s_2\}$, consider the computation trees starting at that state, and for each tree, check whether the given formula holds on it or not.

i. $\mathbf{AG}\, a$

ii. $\mathbf{EG}\, a$

iii. $\mathbf{AF}\,\mathbf{G}\, b$

iv. $\mathbf{AF}\,\mathbf{EG}\, b$

v. $\mathbf{EF}\,\mathbf{G}\, b$

vi. $\mathbf{EX}\, b$

vii. $\mathbf{EG}\,\mathbf{F}\, a$

viii. $\mathbf{A}(b\,\mathbf{U}\,a)$

ix. $\mathbf{A}(a\,\mathbf{U}\,b)$

x. $\mathbf{E}(a\,\mathbf{U}\,b)$

(10 points)

5.) Decision procedures

(a) Consider the following formula in propositional logic; is it satisfiable? If yes, provide a satisfying assignment, if not, give the reasoning that leads to this conclusion.

$$\begin{aligned}(x_1 \vee x_2) \;\wedge\; (x_3 \vee x_4) \;\wedge\; (x_5 \vee x_6) \wedge \\ (\neg x_1 \vee \neg x_3) \;\wedge\; (\neg x_1 \vee \neg x_5) \;\wedge\; (\neg x_2 \vee \neg x_4) \;\wedge\; (\neg x_2 \vee \neg x_6) \wedge \\ (\neg x_3 \vee \neg x_5) \;\wedge\; (\neg x_4 \vee \neg x_5) \;\wedge\; (x_6 \vee \neg x_5 \vee x_1)\end{aligned} \tag{1}$$

(4 points)

(b) Consider the following formulas in Equality Logic and Equality Logic with Uninterpreted Functions (EUF); are they satisfiable? If yes, provide a satisfying assignment over integers, if not, give the reasoning based on equivalence classes that leads to this conclusion.

$$i = j \ \wedge \ j = k \ \wedge \ k = l \ \wedge \ l \neq m \ \wedge \ l \neq n \ \wedge \ m = n \ \wedge \ o \neq p \wedge o = q \tag{2}$$

$$i = j \ \wedge \ j = k \ \wedge \ k = l \ \wedge \ l \neq n \ \wedge \ m = n \ \wedge \ g(i) \neq g(m) \ \wedge \ f(i) \neq f(l) \tag{3}$$

(2 points)

(c) Construct an Ordered Binary Decision Diagram that encodes the following Boolean formula:

$$(x_1 \oplus x_2) \wedge (x_1) \tag{4}$$

(where $\oplus$ is the exlusive or operator).

Ilustrate the construction steps and not just the final result!

(4 points)