6.0 ECTS/4.5h VU Programm- und Systemverifikation (184.741)
16 June 2023

Kennzahl (study id)	Matrikelnummer (student id)	Familienname (family name)	Vorname (first name)	Platz (seat)

1.) Coverage

Consider the following program fragment and test suite:

```
int fib (unsigned n) {
1 unsigned a = 0;
2 unsigned b = 1;
3 unsigned c;
4 unsigned i = 2;
5 while ((i <= n) || (n == 0)) {
6   c = a + b;
7   a = b;
8   b = c;
9   i = i + 1;
10  if (n == 0)
11      return 0;
12 }
13 return b;
}
```

Input (n)	Output
0	0
1	1

(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)

1	2	3	4	5	6	$\sum$
/18	/10	/10	/09	/10	/03	/60

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

Indicate (✓) which of the following coverage criteria are satisfied by the test-suite above (here, the parameter `n` of the function does not constitute a definition, and the `return` statements are a c-uses):

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

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

(7 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-p-uses/some-c-uses

Input (`n`)	Output

MC/DC

Input (`n`)	Output

(4 points)

(d) **Mutation Testing**

Assume that the assignment `i = 2;` is changed to `i = 1;`. Provide a test case that **strongly kills** the resulting mutant (i.e., a test case for which the mutant provides a return value different from the one provided by the original program and specified by the test case.)

Test Case

Input (`n`)	Output

(2 points)

2.) Hoare Logic

Prove the Hoare Triple below. Assume that the domain of all variables in the program are the natural numbers including 0, i.e., $\mathtt{i}, \mathtt{n}, \mathtt{s} \in \mathbb{N}_0$. 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!

$\{\mathsf{true}\}$

```
if (s % 2 == 0) {

  s = s + 1;

} else {

  skip;

}

i = 0;

while (i != n) {

  i = i + 1;

  s = s + i;

}
```

$\{(\mathsf{s} \geq \mathsf{n} + 1)\}$

(10 points)

3.) Invariants Consider the following program, where `i`, `s`, and `n` are natural numbers in $\mathbb{N}_0$ (i.e., including zero).

```
i = 0;
s = 1;
while (i != n) {
  i = i + 1;
  s = s + i;
}
```

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 `i`, `s`, and `n` before and after the loop body demonstrating that the Hoare triple

$$\{P \wedge B\} \quad \texttt{i} = \texttt{i} + 1;\ \texttt{s} = \texttt{s} + \texttt{i}; \quad \{P\}$$

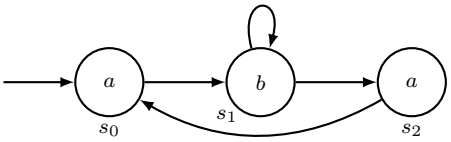
  (where B is `(i != n)`) does not hold.
- Otherwise, provide values of `i`, `s`, and `n` that correspond to a reachable state showing that the formula is *not* an invariant.

$(s \geq 2 \cdot i)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(s \neq i)$	☐ Inductive Invariant	☐ Non-inductive Inv.	☐ Neither
Justification:			
$(2 \cdot i < s)$	☐ 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. $\mathbf{EG}\, a$

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

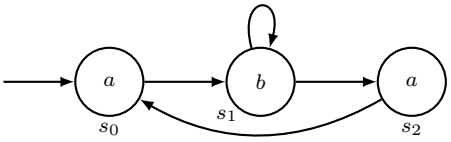
iii. $\mathbf{A}(a \wedge \mathbf{X}\, b)$

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

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

(5 points)

(b) Consider the following Kripke Structure with initial state s_0:



Use the **tableaux algorithm** for CTL from the lecture to compute the sets of states in which the following formula (and its subformulas) hold!

- For every subformula, compute the states for which it holds!
- For fixpoints, list every step of the computation!

$$\mathsf{EG}\,(\mathsf{EX}\,a)$$

(4 points)

5.) Decision procedures

(a) Consider the following formula in propositional logic; is it satisfiable?

- If yes, **provide <u>all</u> satisfying assignments** and **explain how you arrived at that number**
- if not, **provide the steps of the CDCL algorithm that led to this conclusion:**
  - illustrate the conflict graphs for the relevant implication levels and
  - provide the learned clauses.

$$
\begin{aligned}
&(\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (x_2 \vee x_3) \wedge \\
&\qquad (\neg x_3 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge (\neg x_4 \vee \neg x_5) \wedge (x_4 \vee x_5) \wedge \\
&\qquad (\neg x_5 \vee \neg x_6) \wedge (x_5 \vee x_6) \wedge (\neg x_6 \vee \neg x_7) \wedge (x_6 \vee x_7) \wedge \\
&\quad (\neg x_1 \vee \neg x_6 \vee x_7) \wedge (\neg x_1 \vee \neg x_7 \vee x_6) \wedge (\neg x_6 \vee \neg x_7 \vee x_1) \wedge (x_6 \vee x_7 \vee x_1)
\end{aligned}
$$

(6 points)

(b) Provide a BDD for the following Boolean formula such that the size of the BDD is at most polynomial in the number of variables:

$$(\neg(a \Leftrightarrow b)) \wedge (\neg(b \Leftrightarrow c)) \wedge (\neg(c \Leftrightarrow d))$$

Note: You don't have to construct the BDD step-by-step; the final result is sufficient.

(4 points)

6.) General Questions

(a) Indicate whether the following statements are true or false!

Statement	**True**	**False**
Any assertion implied by an (not necessarily inductive) invariant of a program is also an invariant.	◯	◯
Any CTL formula that only contains universal path quantifiers (**A**) can be reformulated as an eqivalent LTL property.	◯	◯
If a program terminates on all inputs, statement coverage can always be achieved.	◯	◯

(3 points)