VU Advanced Software Engineering WS2025 - Test 2, Group B	Duration: 90min TBD
Last name: First name:	Student ID:

This test covers topics from Symbolic Execution, Dynamic Symbolic Execution, Fuzzing, and Advanced Testing Techniques.

This is just an example test.

1. Single Choice Questions (12 points)

In the following, you find some single-choice questions. You get +2 points if you check exactly one answer and this answer is correct.

a) In Symbolic Execution, which statement about path conditions is correct?

- ☐ Path conditions are always satisfiable for any execution path.
- ☐ Path conditions accumulate constraints from branch conditions along the execution path.
- ☐ Path conditions are only used in Dynamic Symbolic Execution.
- ☐ Path conditions cannot contain symbolic variables.

b) Dynamic Symbolic Execution differs from pure Symbolic Execution in that it:

- ☐ cannot handle loops.
- ☐ uses concrete values to handle operations that are difficult to reason about symbolically.
- ☐ is always more precise than Symbolic Execution.
- ☐ does not generate path conditions.

c) In greybox fuzzing, the test suite is updated when:

- ☐ any new input is generated.
- ☐ an input covers new program paths or edges.
- ☐ an input causes a program crash.
- ☐ the fuzzer runs for more than 1 hour.

d) Metamorphic testing is most useful when:

- ☐ we have a perfect test oracle.
- ☐ the program has no loops.
- ☐ we lack a test oracle but can define input-output relations.
- ☐ differential testing is not applicable.

e) For differential testing to be effective, the multiple implementations being compared should:

- ☐ have identical performance characteristics.
- ☐ implement the same specification but have independent implementations.
- ☐ use the same programming language.
- ☐ produce byte-identical outputs.

f) Input prediction in fuzzing aims to:

- ☐ randomly generate new inputs.
- ☐ predict which inputs will crash the program.
- ☐ systematically generate inputs that flip specific branch conditions.
- ☐ minimize the number of test cases.

g) In floating-point arithmetic according to IEEE 754, which statement is correct?

- ☐ Floating-point addition is associative.
- ☐ The expression `x + y - y` always equals `x`.
- ☐ Rounding errors accumulate during computations.
- ☐ All real numbers can be represented exactly.

h) Affine arithmetic improves upon interval arithmetic by:

- ☐ eliminating all over-approximations.
- ☐ tracking linear correlations between variables.
- ☐ providing exact results for all nonlinear operations.
- ☐ requiring less computational resources.

i) When analyzing rounding errors in floating-point programs, which approach provides guaranteed bounds for all possible inputs?

- ☐ Dynamic analysis with concrete execution.
- ☐ Testing with random inputs.
- ☐ Static analysis using interval arithmetic.
- ☐ Profiling the program with typical workloads.

j) For the affine form $\hat{x} = 5 + 3\epsilon_1 + 2\epsilon_2$ where $\epsilon_i \in [-1, 1]$, what is the corresponding interval?

- ☐ $[0, 10]$
- ☐ $[0, 5]$
- ☐ $[2, 8]$
- ☐ $[3, 7]$

k) Which of the following is NOT a special value in IEEE 754 floating-point standard? (1 point)

- ☐ `NaN` (Not a Number)
- ☐ `+`∞ and `-`∞
- ☐ Subnormal numbers
- ☐ Complex numbers

2. Symbolic Execution with Loops (26 points)

a) Perform Symbolic Execution on function `check_loop` for the path where the loop executes exactly once. Use symbolic variables A and B for parameters `a` and `b`. Complete the table and provide the final path condition. (14 points)

```
void check_loop(int a, int b) {
    int x = 0;
    while (a > x) {
        x = x + b;
        a = a - 1;
    }
    assert(x <= a + 10);
}
```

Path	**Symbolic map**	**Path condition**
entry	$a \to A, b \to B$	true

b) Perform Symbolic Execution on function `branches` for the path: entry $\to$ 1 $\to$ 2 $\to$ 4 $\to$ 6 (skipping line 3 and 5). Provide symbolic map and path condition at the assertion. What does the analysis reveal about this path? (12 points)

```
void branches(int x, int y) {
1    if (x < 5) {
2        y = y * 2;
3    } else {
4        y = y + 1;
5    }
6    assert(y > 0);
}
```

3. Dynamic Symbolic Execution (18 points)

Context: Perform Dynamic Symbolic Execution on function `compute` using a **short-first** search strategy. Use initial concrete values: $x = 2, y = 1$. Handle the `abs` function by replacing it with the concrete value during execution. Complete the table. (14 points)

```
int compute(int x, int y) {
    assume(x > 0 && y > 0);
    int z;
    if (x == y) {
        z = x + y;
    } else {
        z = abs(x - y);
    }
    assert(z >= x);
}
```

Path ID	Parent	Prefix	Input	Path condition
1	-		x=2, y=1	

b) What does the analysis reveal about the assertion's safety? (4 points)

4. Fuzzing (26 points)

a) Given function `test_func` and initial inputs from a greybox fuzzer run, compute the resulting test suite. Provide the path coverage. (10 points)

```
int test_func(int a, int b) {
1    if (a > 10) {
2        if (b < 0) {
3            return a + b;
4        }
5        return a - b;
6    }
7    if (b > 5) {
8        return b * 2;
9    }
10   return 0;
}
```

Initial inputs:

- I_1: $a = 5, b = 3$
- I_2: $a = 15, b = -2$
- I_3: $a = 8, b = 7$

b) Provide two additional inputs that the fuzzer would add to the test suite and two that it would discard. Justify. (6 points)

c) Perform fuzzing with input prediction starting with inputs I_1 and I_2 to flip condition `b < 0` by predicting value for `b`. Provide the cost expression, cost values, and linear cost function. (6 points)

d) Determine the next predicted input. Does it successfully flip the condition? (4 points)

5. Advanced Testing Techniques (18 points)

a) For the function `max3` below, which returns the maximum of three integers, design a metamorphic testing approach. Define formally one metamorphic input relation R^I and matching output relation R^O. (10 points)

```
int max3(int a, int b, int c) {
    // Returns maximum of a, b, c
}
```

b) You want to test a C compiler using differential testing with three different compilers (GCC, Clang, MSVC). Explain how you would set up the differential testing and what challenges might arise. (4 points)

c) Explain one advantage of combining Abstract Interpretation with Symbolic Execution for verifying programs with complex loops compared to using Symbolic Execution alone. (4 points)