

CS130: Software Engineering (Fall 2019)

Assignment 2

Due Date: December 6, 11:55 PM

Late Policy: There is a 10% penalty per every late day. Late assignments are not accepted after 48 hours. Once a solution is released, there's no credit.

Submission Instruction: Please submit **a single typed-out** report file (.pdf, .doc, .docx) to CCLE before the deadline.

Q1: Symbolic Execution [25pts]

Alice wrote the following Java function and would like to reason about test adequacy of her function. Please perform symbolic execution on the Java function first and then answer the following sub-questions about testing.

```
static int compute(int x, int y, int z) {
S1: if (x < y) {
S2:     if (y >= z) {
S3:         z = 135;
        } else {
S4:         z = x;
S5:         y = z;
        }
    } else {
S6:     z = -1 * y;
    }

S7: if (z > 0 && z < x + y) {
S8:     x = z + 10;
    } else {
S9:     y = z - 10;
    }
S10: return x + y + z;
}
```

(a) Please identify the feasibility of each path, their corresponding path condition and effect by filling in the following table. The first entry is done for you as an example
[15 pts]

S1	S2	S7	Path as statement labels	Feasible?	Path condition	Effect
T	T	T	{S1, S2, S3, S7, S8, S10}	Yes	$X < Y \ \&\& \ Y \geq Z \ \&\& \ X + Y > 135$	$x' = 145$ $y' = y$ $z' = 135$
T	T	F				
T	F	T				
F	T	T				
T	F	F				
F	F	T				
F	T	F				

F	F	F				
---	---	---	--	--	--	--

(b) Please cluster the following test cases based on whether the test cases exercise the same path. In your answer, please identify each path using statement labels.

Example: if Si, Sj and Sk are the statements corresponding to path P1, you must represent path P1 as $P1 = \{Si, Sj, Sk\}$. If test cases Ti, Tj and Tk cover the same path P1, then you must represent your answer as the following: **{Ti, Tj, Tk} covers path {Si,Sj,Sk}. [10]**

T1: $x = 10$ AND $y = 7$ AND $z = 5$

T2: $x = 15$ AND $y = 11$ AND $z = 7$

T3: $x = 3$ AND $y = 4$ AND $z = 9$

T4: $x = 100$ AND $y = -190$ AND $z = 10$

T5: $x = 99$ AND $y = 100$ AND $z = 47$

Q2: Test Coverage [25pts]

In the following Java program, each statement is marked with labels:

```

int coverage(int a, int b, int x, int y){
s1    int value = 0;

s2    if (a > x) {
s3        value = 1;
        } else {
s4        value = -1;
        }

s5    if (b > y) {
s6        value += 1;
s7    } else if (value * x <= -a) {
s8        value -= 1;
        }

s9    return value;
}

```

- A) Please draw the control flow graph of this Java program. Please annotate each CFG node with the corresponding statement labels **[5pts]**
- B) Please create a minimum number of tests to achieve 100% statement coverage. Please write your tests in the format of JUnit test cases and explain which statements each test covers using the labels. **[5pts]**
- C) Please augment your tests in (b) with a minimum number of additional tests to achieve 100% branch coverage. **[5pts]**
- D) Please augment your tests in (b) and (c) with a minimum number of additional tests to achieve 100% coverage of all feasible paths. **[5pts]**
- E) Identify all infeasible paths (if any) and explain why. **[5pts]**

Q3 Bounded Loop Iteration [20pts]

Consider the following Java program.

```
int loop(int N) {
    Random rand = new Random();
    for(int i=0; i<N; i++)
    {
        for(int j=0; j<Math.pow(2,3*(i+2)); j++)
        {
            if(rand.nextInt() % 2 == 0) //s1
            {
                System.err.println("Reached here!");
            }
        }
    }
}
```

- (a) Given N, how many times is the “if condition” (labelled as s1) examined? Please include your derivation in your answer. **[10pts]**
- (b) How many feasible paths do you have in this program? The answer should only contain N as a variable. **[10pts]**

Q4 Who's the Weakest? [20pt]

Assume that the post-condition of the following Java program is TRUE, meaning that it should terminate without throwing any exceptions. Through the weakest precondition reasoning, your goal is to identify the program inputs and intermediate program states such that the program produces the post-condition of TRUE. While the sub-questions are multiple choice questions, please include derivation of weakest precondition reasoning in your answer.

```
private int getSomeElement(MyObject o, int len, int index){
    int[] s;
    s = new int[len];
    if (o != null) {
P1:        len = o.s;
           s = o.arr;
    } else {
P2:        len= len - index;
    }
P3:        return s[len-5];
    }

    class MyObject{
        int s;
        int[] arr;
    }
}
```

(a) What is the weakest pre-condition at the program point P1?

- A. $o.arr \neq \text{null} \text{ AND } o.s < o.arr.length + 5$
- B. $o.s \geq 5 \text{ AND } o.arr \neq \text{null} \text{ AND } o.s < o.arr.length + 5$
- C. $o.s > 5 \text{ AND } o.arr \neq \text{null} \text{ AND } o.s < o.arr.length + 5$
- D. $o.s \geq 5 \text{ AND } o.arr \neq \text{null} \text{ AND } o.s < o.arr.length + 5 \text{ AND } o.s \neq \text{null}$
- E. None of the above

(b) Which of the following is a precondition at the program point P2 but is NOT the weakest? Choose all that apply.

- A. $\text{index} + 10 \leq \text{len}$ AND $\text{len} < \text{s.length} + \text{index} + 5$
- B. $\text{index} \leq \text{len}$ AND $\text{len} < \text{s.length} + \text{index} + 5$
- C. $\text{index} + 5 \leq \text{len}$ AND $\text{len} < \text{s.length} + \text{index} + 5$
- D. $\text{index} + 5 \leq \text{len}$ AND $\text{len} < \text{s.length}$ AND $\text{index} > -5$
- E. None of the above

(c) What is the weakest pre-condition at the program point P3?

- A. $5 < \text{len}$ AND $\text{len} < \text{s.length} + 5$
- B. $5 \leq \text{len}$ AND $\text{len} < \text{s.length}$
- C. $5 \leq \text{len}$ AND $\text{len} < \text{s.length} + 5$
- D. $5 \leq \text{o.s}$ AND $\text{len} < \text{o.arr.length} + 5$
- E. None of the above

(d) What is the weakest precondition of the entire program (at the beginning of the program)?

- A. $(\text{o} \neq \text{null} \text{ AND } \text{o.s} \geq 0 \text{ AND } \text{o.arr} \neq \text{null} \text{ AND } \text{o.s} < \text{o.arr.length})$
OR $(\text{o} == \text{null} \text{ AND } \text{index} + 5 \leq \text{len} \text{ AND } \text{index} > -5 \text{ AND } \text{len} \geq 0)$
- B. $(\text{o} \neq \text{null} \text{ AND } \text{o.s} \geq 5 \text{ AND } \text{o.arr} \neq \text{null} \text{ AND } \text{o.s} < \text{o.arr.length} + 5)$
OR $(\text{o} == \text{null} \text{ AND } \text{index} \leq \text{len} \text{ AND } \text{len} \geq 0)$
- C. $(\text{o} \neq \text{null} \text{ AND } \text{o.s} \geq 5 \text{ AND } \text{o.arr} \neq \text{null} \text{ AND } \text{o.s} < \text{o.arr.length})$
OR $(\text{o} == \text{null} \text{ AND } \text{index} \leq \text{len} + 5 \text{ AND } \text{index} > -5 \text{ AND } \text{len} \geq 0)$
- D. $(\text{o} \neq \text{null} \text{ AND } \text{o.s} \geq 5 \text{ AND } \text{o.arr} \neq \text{null} \text{ AND } \text{o.s} < \text{o.arr.length} + 5)$
OR $(\text{o} == \text{null} \text{ AND } \text{index} + 5 \leq \text{len} \text{ AND } \text{index} > -5 \text{ AND } \text{len} \geq 0)$ // $\text{len} \geq 0$ should be applied to the first part as well.
- E. None of the above

Q5: Mutants [10pts]

The following function takes an integer array and length of an array as input, and sorts the array in ascending order. In order to assess the adequacy of existing test suites, Tom leverages the mutation analysis to create three variants of the function and see if existing tests can kill such mutants as shown below. *Mutant 1* replaces the constant '1' in the if condition in line 4 with '2'. *Mutant 2* replaces the greater than operator in line 9 with less than. *Mutation 3* deletes the statement in line 14.

```
1 void bubble_sort(int list[], int n)
2 {
3     int i, j, temp;
4     if( n == 0 || n == 1) return; // mutant 1: n == 1 is changed to n == 2
5     for (i = 0 ; i < ( n - 1 ); i++)
6     {
7         for (j = 0 ; j < n - i - 1; j++)
8         {
9             if (list[j] > list[j+1]) // mutant 2: > is changed to <
10            {
11                /* Swapping */
12                temp = list[j];
13                list[j] = list[j+1];
14                list[j+1] = temp; // mutant 3: this statement is deleted
15            }
16        }
17    }
18 }
```

Please answer the following multiple choice questions. Each choice in the questions indicates a test case in the format of (input, expected output). For example, (list = [1,3,2] ^ n=3, list = [1,2,3]) means that given an array [1,3,2] and the array length is 3, the function will sort the array to [1,2,3].

A. [5pts] Please select all tests that kill mutant 1:

- a. (list = [] ^ n=0, list=[])
- b. (list = [1] ^ n=1, list=[1])
- c. (list = [2,1] ^ n=2, list=[1,2])
- d. (list = [1,3,2] ^ n=3, list=[1,2,3])

B. [5pts] Please select all tests that kill Mutant 1, Mutant 2, Mutant 3:

- a. (list=[-1,-4] ^ n=2, list=[-4,-1])
- b. (list=[1,2] ^ n=2, list=[1,2])
- c. (list=[3,2,1,0] ^ n=4, list=[0,1,2,3])
- d. (list=[] ^ n=0, list=[])