CSE 4321: Software Testing and Maintenance
Fall 2020

Final Exam

12/10/2020

1. This is an open book exam.
2. This exam consists of 5 problems and has a total of 90 points.
3. The length of this exam is 90 minutes, from 11.00am to 12.30pm. Use your time efficiently by working on the easier problems first.
4. Your answers must be submitted to Canvas by 12.40pm. You are suggested to start submission at 12.30pm so that there will be time to deal with any technical issues.
5. If you use the back of the exam sheets or if you use additional sheets, please indicate so. Write your name on the additional sheets.

Name: ____
First Last

Problem 1: (20 Points)

1. (2 points) (True or False) Software maintenance must work within the parameters and constraints of an existing system.

2. (2 points) (True or False) Reverse engineering is the process of analyzing the source code to create system representations at higher levels of abstraction.

3. (2 points) (True or False) Configuration management allows different releases to be made from the same code base.

4. (2 points) (True or False) In a version model, the product space describes the structure of a software product while taking versioning into account.

5. (2 points) (True or False) In extensional versioning, the versions of a software product are implicit and constructed on demand.

6. (2 points) (True or False) A variant is a version intended to co-exist with its predecessor.

7. (2 points) (True or False) The only purpose of code review is to detect faults that may exist in the source code.

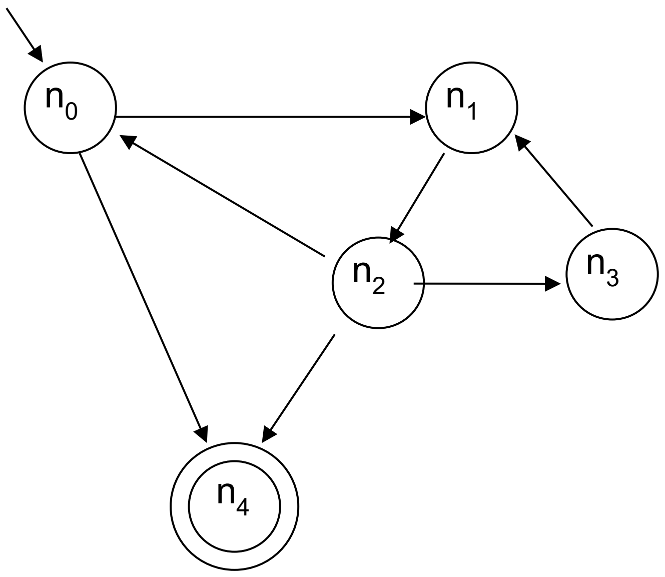
8. (2 points) (True or False) Management should not use code review to measure the performance of a software developer.

9. (2 points) (True or False) Refactoring changes the external behavior of a software system, without changing its internal structure, to make the software system easier to understand and maintain.

10. (2 points) (True or False) Refactoring should be performed when it is close to a project deadline.

Problem 2: (10 + 5 = 15 points)

Consider the following control-flow graph, where node n_0 is the initial node and node n_4 is the final node.



(a) Identify all the prime paths in the graph.
(b) Identify a test path set (i.e., one or more test paths) that achieves prime path coverage for the graph.

(This blank page provides additional space for Problem 2.)

Problem 3: (5 + 5 + 5 + 5 + 5 = 25 points)

Consider predicate $p = a \wedge b \wedge (\neg c \vee d)$. Answer the following questions:

(a) Compute (and simplify) the conditions under which clause *c* determines predicate *p*.
(b) Write the complete truth table for all the clauses, with rows labeled starting from 1. Note that you should include a column for the condition under which clause *c* determines the predicate, and also a column for the predicate itself.
(c) Identify all pairs of rows from your table that satisfy General Active Clause Coverage (GACC) with respect to clause *c*.
(d) Identify all pairs of rows from your table that satisfy Correlated Active Clause Coverage (CACC) with respect to clause *c*.
(e) Identify all pairs of rows from your table that satisfy Restricted Active Clause Coverage (RACC) with respect to clause *c*.

(This blank page provides additional space for Problem 3.)

Problem 4: (15 Points)

Consider the following method *cal*:

```
/*
 * Calculate the number of days between two given days in the same year. Assume
 * all the parameters take valid values. In particular, they satisfy the following
 * preconditions: day1 and day2 must be in the same year
 *                    1 <= month1, month2 <= 12
 *                    1 <= day1, day2 <= 31
 *                    month1 <= month2
 *                    the range for year: 1 … 10000
 */
public static int cal (int month1, int day1, int month2, int day2, int year) {
   int numDays;
   if (month2 == month1) // in the same month
     numDays = day2 – day1;
   else {
     // skip month 0
     int daysIn[] = {0, 31, 0, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
     // are we in a leap year
     int m4 = year % 4;
     int m100 = year % 100;
     int m400 = year % 400;
     if ((m4 != 0) || ((m100 == 0) && (m400 != 0)))
        daysIn[2] = 28;
     else
        daysIn[2] = 29;
      // start with days in the two months
      numDays = day2 + (daysIn[month1] – day1);
      // add the days in the intervening months
      for (int i = month1 + 1; i <= month2 – 1; i ++)
          numDays = daysIn[i] + numDays;
   }
   return (numDays);
}
```

Assume that a mutant of the above method is created by replacing "**if** ((*m4* != 0) || ((*m100* == 0) && (*m400* != 0)))" with "**if** ((*m4* != 0) || ((*m100* != 0) && (*m400* != 0)))". Provide reachability conditions, infection conditions, propagation conditions and test case values to kill this mutant.

(This blank page provides additional space for Problem 4.)

Problem 5: (15 points)

Suppose program P has been executed against a test suite T consisting of six tests, t1, t2, t3, t4, t5 and t6. A total of six entities are covered by the tests as shown in the following table: 0 (or 1) in a column indicates that the corresponding entity is not covered (or covered). The entities could be basic blocks in the program, functions, def-uses, or any other testable elements of interest. Follow procedure CMIMX to find the minimal cover set for the six entities. Note that intermediate steps must be shown in order to get full credits.

	1	2	3	4	5	6
t1	0	1	0	0	0	1
t2	1	0	1	0	0	1
t3	0	0	0	1	0	1
t4	0	0	1	0	0	0
t5	0	1	1	0	0	0
t6	0	0	1	0	1	1

(This blank page provides additional space for Problem 5.)