CSCE 747 - Midterm
Name:

This is a 75-minute exam. On all essay type questions, you will receive points based on the quality of the answer - not the quantity.

Make an effort to write legibly. Illegible answers will not be graded and awarded 0 points.

There are a total of 8 questions and 100 points available on the test.

Question 1 - 12 Points

Multiple solutions may apply. Select all that are applicable.

1. A test suite that meets a stronger coverage criterion will find any defects that are detected by any test suite that meets only a weaker coverage criterion
    a. True
    b. False

2. A test suite that is known to achieve Modified Condition/Decision Coverage (MC/DC) for a given program, when executed, will exercise, at least once:
    a. Every statement in the program.
    b. Every branch in the program.
    c. Every LCSAJ in the program.
    d. Every path in the program.

3. Possible sources of information for functional testing include:
    a. Requirements Specification
    b. User Manuals
    c. Program Source Code
    d. Domain Experts

4. Category-Partition Testing technique requires identification of:
    a. Parameter characteristics
    b. Representative values
    c. Def-Use pairs
    d. Pairwise combinations

5. Validation activities can only be performed once the complete system has been built.
    a. True
    b. False

6. Statement coverage criterion never requires as many test cases to satisfy as branch coverage criterion.
    a. True
    b. False

7. Requirement specifications are not needed for generating inputs to satisfy structural coverage of program code.
    a. True
    b. False

8. A system that fails to meet its user's needs may still be:
    a. Correct with respect to its specification.
    b. Safe to operate.
    c. Robust in the presence of exceptional conditions.
    d. Considered to have passed verification.

Question 2 - 12 Points

Consider the following situation:
After *carefully and thoroughly* developing a collection of requirements-based tests and running your test suite, you determine that you have achieved only 60% statement coverage. You are surprised (and saddened), since you had done a very thorough job developing the requirements-based tests and you expected the result to be closer to 100%.

1. Briefly describe two (2) things that might have happened to account for the fact that 40% of the code was not exercised during the requirements-based tests.
2. Should you, in general, be able to expect 100% statement coverage through thorough requirements-based testing alone (why or why not)?
3. Some structural criteria, such as MC/DC, prescribe obligations that are impossible to satisfy. What are two reasons why a test obligation may be impossible to satisfy?

Question 3 - 6 Points

In class we discussed the importance of defining a test case for each requirement. What are the two primary benefits of defining this test case?

Question 4 - 15 Points

The airport connection check is part of a travel reservation system. It is intended to check the validity of a single connection between two flights in an itinerary.

validConnection(Flight arrivingFlight, Flight departingFlight) returns ValidityCode.

A Flight is a data structure consisting of:

- A unique identifying flight code (string, three characters followed by four numbers).
- The originating airport code (three character string).
- The scheduled departure time (in universal time).
- The destination airport code (three character string).
- The scheduled arrival time (in universal time).

There is also a flight database, where each record contains:

- Three-letter airport code (three character string).
- Airport country (two character string).
- Minimum connection time (integer, minimum number of minutes that must be allowed for flight connections).

ValidityCode is an integer with value 0 for OK, 1 for invalid airport code, 2 for a connection that is too short, 3 for flights that do not connect (arrivingFlight does not land in the same location as departingFlight), or 4 for any other errors (malformed input or any other unexpected errors).

In order to design requirements-based test cases, perform category-partition testing using this specification for the validConnection function.

1. Identify parameters.
2. Identify categories for each parameter.
3. Identify representative values (choices) for each category.

Question 5 - 15 Points (5x3)

For the following function,

```
int findMax(int a, int b, int c)
{
   int temp;
   if (a>b)
      temp=a;
   else
      temp=b;

   if (c>temp)
      temp = c;
   return temp;
}
```

a. Draw the control flow graph for the program.

b. Develop test input that will provide statement coverage. (Input output pairs will be fine.)

c. Develop test input that will provide branch coverage.

d. Develop test input that will provide full-path coverage.

e. Modify the program to introduce a fault so that you can demonstrate that even achieving full path coverage will not guarantee that we will reveal all faults. Please explain how this fault is missed in your example.

Question 6 - 12 Points

Identify all DU Pairs in the following code:

```
1.
2.  /* External file hex_values.h defined Hex_Values[128]
3.  * with value 0 to 15 for the legal hex digits
4.  * and value -1 for each illegal digit including special
5.  * characters */
6.
7.  #include “hex_values.h”
8.  /** Translate a string from the CGI encoding to plain
9.  * acsii text. ‘+’ becomes space, %xx becomes byte with hex
10. * value xx, other alphanumeric characters map to themselves
11. * Returns 0 for success, positive for erroneous input.
12. * 1 = bad hexadecimal digit.
13. */
14. int cgi_decode(char *encoded, char *decoded){
15.             char *eptr = encoded;
16.             char *dptr = decoded;
17.             int ok = 0;
18.             while(*eptr){
19.                     char c;
20.                     c = *eptr;
21.
22.                     if(c ==’+’){   /* Case 1: ‘+’ maps to blank*/
23.                             *dptr = ‘ ‘;
24.                     } else if(c == ‘%’){  /* Case 2: ‘%xx’ = char xx*/
25.                             int digit_high = Hex_Values[*(++eptr)];
26.                             int digit_low = Hex_Values[*(++eptr)];
27.                             if(digit_high == -1 || digit_low == -1){
28.                                     /* *dptr=’?’ */
29.                                     ok = 1; /* Bad return code */
30.                             }else{
31.                                     *dptr = 16 * digit_high + digit_low;
32.                             }
33.                     }else{ /*Case 3: All other chars map to themselves*/
34.                             *dptr = *eptr;
35.                     }
36.                     ++dptr;
37.                     ++eptr;
38.             }
39.             *dptr = ‘\0’;
40.             return ok;
41.     }
```

Question 7 - 16 Points (4x4)

In a directed graph with a designated exit node, we say that a node m post-dominates another node n, if m appears on every path from n to the exit node.

Let us write *m pdom n* to mean that m post-dominates n, and *pdom(n)* to mean the set of all post-dominators of n, i.e., *{m | m pdom n}*.

Answer the following, providing justification for each:

1. Does *b pdom b* hold true for all b?
2. Can both *a pdom b* and *b pdom a* hold true for two different nodes a and b?
3. If both *c pdom b* and *b pdom a* hold true, what can you say about the relationship between c and a?
4. If both *c pdom a* and *b pdom a* hold true, what can you say about the relationship between c and b?

Question 8 - 12 Points

In class, we discussed various forms of oracles – such as a model, a second implementation, properties, self-checks, a team of experts, etc. Provide a comparative analysis of three different kinds of oracles of your choice, defining what they are and addressing their strengths and weaknesses with respect to key attributes relevant to the verification process (e.g., cost, accuracy, completeness).