Department of Computer Science
Purdue University, West Lafayette

Fall 2006: CS 490M Software Testing
Final Examination: Solutions

Monday December 11, 2006. 8:15am–10am. LWSN B134

Maximum points: 100 (To be scaled down to 25 during final course grading.)

Name: Aditya Mathur

All answers are in red.

Answer all questions in the space provided. When needed, use the empty space on the back of each sheet.

Q 1 Select the correct option. *10 points*

(i) Acceptance testing is usually done by application developers. — No

(ii) A test generated using covering arrays is adequate with respect to decision coverage. — No

(iii) Decision coverage subsumes statement coverage. — Yes

(iv) The characterization set of a minimal and connected FSM is minimal and contains no redundant string. — Yes

(v) Chow's W-method generates a minimal set of tests required to detect all faults using Chow's fault model. — No

(vi) Tests generated by using Chow's W-method are adequate with respect to decision coverage. — No

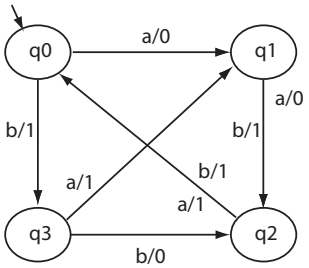
(vii) A pairwise design is aimed at generating tests to detect errors due to the interaction of two program variables. — Yes

(viii) An orthogonal array $OA(4, 3, 2, 2)$ contains three runs for four factors, each having two values and has a strength of two. — No

(ix) For the same set of parameters, a covering design usually contains fewer runs than that based on an orthogonal array. — Yes

(x) When applied to a program, a mutation operator generates one or more mutants. — No

Q 2 (a) Suppose FSM $M = (X, Y, Q, q_0, \delta, O)$ is such that $X = \{a, b\}$, $Y = \{0, 1\}$, $Q = \{q_0, q_1, q_2, q_3\}$, q_0 is the start state, and the transition and output functions are as shown below. Note that edges $q_1 - q_2$ and $q_2 - q_0$ have multiple labels. Construct the k-equivalence partition tables and compute the characterization set W for M. *50 points*



(i) Begin by filling the following state transition and output table for M.

Current state	Output		Next state	
	a	b	a	b
q_0	0	1	q_1	q_3
q_1	0	1	q_2	q_2
q_2	1	1	q_0	q_0
q_3	1	0	q_1	q_2

(ii) Next, prepare to create the P_1 table by filling in entries in the following table. Indicate the 1-equivalence partition by entering values for Σ. Separate the states in the different equivalence classes by suitably placing a horizontal line.

Σ	Current state	Output		Next state	
		a	b	a	b
1	q_0	0	1	q_1	q_3
	q_1	0	1	q_2	q_2
2	q_2	1	1	q_0	q_0
3	q_3	1	0	q_1	q_2

State transition and output table for M; Groupings indicated by horizontal lines at the appropriate point.

(iii) Now remove the **Output** column from the table above and rewrite it as shown below. Note that states under the **Next State** column will now be shown as $q_{i\,j}$ to indicate state i and group j. Don't forget to fill in the Σ column and place the horizontal line separating the groups.

Σ	**Current state**	**Next state**	
		a	b
1	q_0	$q_{1\,1}$	$q_{3\,3}$
	q_1	$q_{2\,2}$	$q_{2\,2}$
2	q_2	$q_{0\,1}$	$q_{0\,1}$
3	q_3	$q_{1\,1}$	$q_{2\,2}$

P_1 table.

(iv) Next, fill in the P_2 and P_3 tables, but only if necessary.

Σ	**Current state**	**Next state**	
		a	b
1	q_0	$q_{1\,2}$	$q_{3\,4}$
2	q_1	$q_{2\,3}$	$q_{2\,3}$
3	q_2	$q_{0\,1}$	$q_{0\,1}$
4	q_3	$q_{1\,1}$	$q_{2\,2}$

P_2 table.

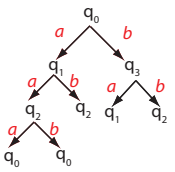
Partitioning is complete and hence no need for P_3.

(v) Derive the characterization set for M from the equivalence tables constructed in (a). *10 points*

W-set={a, b, aa}. Note that other answers are also possible. a distinguishes q_3 and q_2 from q_0 and q_1. b distinguishes q_2 from q_3. aa distinguishes q_0 from q_1.

(b) Construct the testing tree for M. *10 points*

Testing tree for M:



(c) Find the transition cover set P from the testing tree in (b). *10 points*

$P = \{\epsilon, a, b, aa, ab, aaa, aab, ba, bb\}$

Q3(a) Consider the following program that contains an error as indicated. Develop, if possible, an MC/DC adequate test set for the following program that will NOT detect the error. If not possible, then explain why. *15 points*

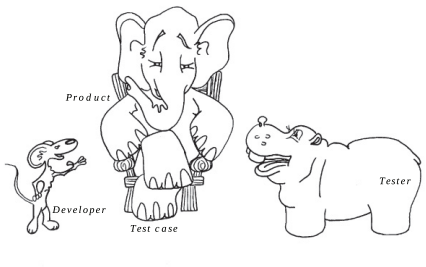
```
1   begin
2     int x, y; float z;
3     input (x, y);
4     z=0;
5     if (x! =0)
6       z=z+y;
7     else z=z-y;
8     if (y! =0) ← This condition should be (y! =0 and x! =0) to avoid division by 0.
9       z=z/x;
10    else z=z*x;
11    output(z);
12  end
```

Fill in as many test cases as needed for MC/DC adequacy. Add additional rows if you need more than four or simply leave rows blank if you need less than four. Note that you will develop the test set based on the given, not the correct, program. The **Expected** value of z is the one generated by the *correct* program. The **Actual** value of z, or an exception, is generated by the program under test.

Test case	x	y	**Expected** z	**Actual** z	**Error detected?**
1	0	0	0	0	No
2	1	1	1	1	No

(b) Is a def-use adequate test set for the above program guaranteed to reveal the error? Provide brief reason to support your answer. *5 points*

Yes. Consider the following def-use pair: definition of z at line 7 and use of z at line 9. The only way to cover this def-use pair is to execute the program such that $x = 0$ and $y = 1$ (any nonzero value of y will suffice). However, this test case forces a divide-by-zero exception at line 9 and hence the error is revealed. The divide-by-zero exception would not occur in the correct program as line 9 is protected by the condition `x!=0`.



♠ "*Elephants do not forget their test cases*", Kevin McCarthy. ♠

♣♭ Have a great Holiday Season. ♣♯