Department of Computer Science
Purdue University, West Lafayette

Fall 2006: CS 490M Software Testing
Midterm Examination

Friday October 6, 2006 9:30pm-11am, LWSN B134

Maximum points: 100 (To be scaled down to 15 during final course grading.)

Name: Aditya Mathur

All answers are in red.

Q 1 Application P is under test. The input domain of P can be partitioned into two subsets: $\mathcal{V}$ consisting of input values that satisfy the constraints on the input variables and $\mathcal{N}$ consisting of input values that do not satisfy the constraints on the input variables.

(a) What type of testing selects test inputs mostly from $\mathcal{N}$?

○ Robustness testing ○ Stress testing

1 point

(b) Imagine, and list, at least one aspect of P that might need stress testing ?

1 point

Handling of incoming requests that might be held in a buffer by P prior to processing. In this case, stress testing will aim at detecting errors in handling buffer overflows and possibly unacceptable performance degradation when "too many" transactions arrive at P in a short interval.

(c) Regression testing is most likely to be used in which phase of the software lifecycle for P ? (Select one)

○ Design ○ Coding ○ Unit testing ○ Maintenance

1 point

(d) Offer one reason why robustness testing might be recommended for P despite the fact that P satisfies all requirements as determined through functional test.

1 point

Incompleteness and ambiguity in requirements as well as errors made by the programmer in handling unexpected values, make robustness testing necessary.

(e) Suppose that P contains no loops but has $n > 0$ `if-then` statements arranged in sequence (not nested). Assume that the length of a path is measured in terms of the number of blocks, not necessarily distinct, along the path. Answer the following.

8 points

Maximum number of distinct paths in $P = 2^n$

Number of basic blocks in $P = 2n + (2)$

Length of the longest path in $P = 2n + (2)$

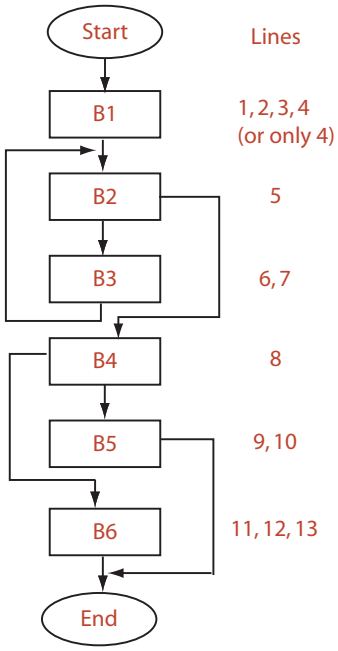
Length of the shortest path in $P = n + (2)$

(2) is necessary when the *start* and *exit* are treated as separate blocks.

(f) Consider the following Java code that contains one method. Draw the control flow graph (CFG) of this procedure. Indicate block numbers, e.g. B_1, B_2, and so on, on your CFG. Indicate the contents of each block by specifying, in a separate table, the line numbers that fall in each block.

16 points

```
1  Button save=new Button ("Save");
2  Button read=new Button ( "Read");
3  int [ ] A={-19, 20, 34, 0, 32, -87};
4  public Button findButton(int index){
5    while(A[index]<0){
6      index++;
7    }
8    if(index==0) {
9      return(save);
10   }
11   else{
12     return(read);
13   }
```



Q 2 The *ls* command in Unix/Linux is to display the contents of a directory. The format of the *ls* command is:

ls options [*pathnames*]

where *options* is a list of options and [*pathnames*] is an optional list of directory names the contents of which are to be displayed. While there is a rather long list of options available, we restrict to two: `-a` and -C. The `-a` option "Shows you all files, even files that are hidden (these files begin with a dot.)." The -C option produces "Multi-column output with entries sorted down the columns. Generally this is the default option."

Your task is to apply uni-dimensional equivalence partitioning and boundary value analysis to generate test cases to test an implementation P of the *ls* command. Do not ignore the fact that files and directories also serve as input to P. While deriving the input data, you do not need to list the contents of a file or directory; simply indicate what should be the properties of the file/directory.

Label your equivalence classes as E_1, E_2, and so on. List the equivalence classes, and a sample test input for each class, in a suitably chosen tabular format. Mark the tests so that it is clear which tests you derived using equivalence partitioning and which ones using boundary value analysis.

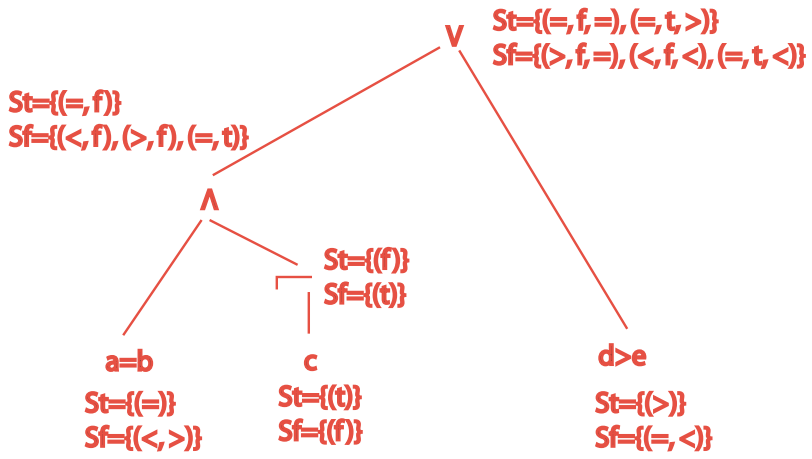
20 points

Eq. Class	Description	Test
Based on options		
E1*	ls commands with no options	ls
E2	with -C option	ls -C
E3	with -a option	ls -a
E4	with illegal options	ls -+
Based on pathnames		
E5	Commands with valid path names	
E6	with invalid pathnames	
E7*	with empty pathname	
E8*	with very long valid path names	
Based on directory contents		
E9	Directories with only hidden files	
E10	Directories with no hidden files	
E11	Directories with a mix of hidden and other files	
E12*	Huge directory	
E13*	Empty directory	

* From BVA.

Q 3(a) Consider the following predicate $p : a = b \wedge \neg c \vee d > e$, where a, b, d, e are integers and c is Boolean. Derive a BRO constraint set S for p. Clearly show the tree used to derive S.

12 points



(b) Use S to derive a sample set T of test inputs.

4 points

Test	Constraint	a	b	c	d	e	p	p_1	p_2
t_1	$(>, f, =)$	3	2	f	3	3			
t_2	$(<, f, <)$	2	4	f	3	4			
t_3	$(=, t, <)$	1	1	t	2	5			
t_4	$(=, f, =)$	5	5	f	0	0	t	f	f
t_5	$(=, t, >)$	4	4	t	4	2			

(c) Show that there are one or more test cases in T that reveal the errors in the following implementations of p:

$p_1 : a = b \wedge c \vee d > e$

$p_2 : a = b \wedge c \vee d < e$

4 points

Test t_4 reveals the errors in p_1 and p_2.

Q 4 Let FSM $M = (X, Y, Q, q_0, \delta, O)$ be minimal, completely specified, connected, and deterministic. Let i/o denote a label along a transition in M, where i belongs to the input alphabet X and o to the output alphabet O. Recall that δ and O are, respectively, the transition and output functions, and q_0 the start state. Let TT denote the transition tree, P the transition cover set, and W the characterization set for M. FSM M' is an implementation of M and contains the same number of states as in M and has the same input and output alphabets as M. Answer the following questions.

(a) Let q_1 and q_2 be two states in Q. Suppose that transitions going out of q_1 have exactly the same labels as those going out of q_2. For example, if $a/0$ is a label on a transition going out of q_1, then q_2 also has an outgoing transition labeled $a/0$. Are q_1 and $q2$ equivalent ? Provide brief support to your answer.

8 points

No, q_1 and $q2$ are not equivalent. They are 1-equivalent but might not be k-equivalent for $k > 1$. For an example, see states q_0 and q_2 in Figure 5.9 of the textbook.

(b) Suppose FSM M 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 in Figure 1. Construct the k-equivalence partition tables to determine if M is minimal. If it is, then find its characterization set W.

18 points

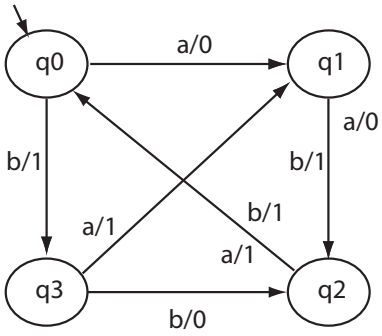


Figure 1 FSM for Q 4(c). Note that edges $q1 - q_2$ and $q_2 - q_0$ have multiple labels.

$P_1 = \{\Sigma_1, \Sigma_2, \Sigma_3\}$, where $\Sigma_1 = \{q_0, q_1\}$, $\Sigma_2 = \{q_2, \}$, $\Sigma_3 = \{q_3\}$

$P_2 = \{\Sigma_1, \Sigma_2, \Sigma_3, \Sigma_4\}$, where $\Sigma_1 = \{q_0\}$, $\Sigma_2 = \{q_1\}$, $\Sigma_3 = \{q_2\}$, $\Sigma_4 = \{q_3\}$

$W = \{a, b, aa\}$ (Other answers are also possible.)

Q 5 You have been provided with a class called Square written in Java. Your job now is to test the methods contained in it using JUnit. Answer the following questions:

(a) What three steps do you follow to create tests that run with JUnit ?

- Create a test class, for example SquareTest
- Import the JUnit headers
- Implement tests and annotate them with @org.junit.Test

(b) How do you validate conditions in your tests ?

By using any of the assert methods provided by JUnit

(c) What feedback does JUnit provide when tests fail ?

- The name of the failed test and also its class
- A message, depending on the assert method used of the form expected:0 <value1> but was:<value2>
- A summary of how many tests were run and how many failed

(d) What feedback does JUnit provide when tests succeed ?

- Time to execute the tests
- OK message
- Number of tests run

(e) Name one advantage of using JUnit for testing

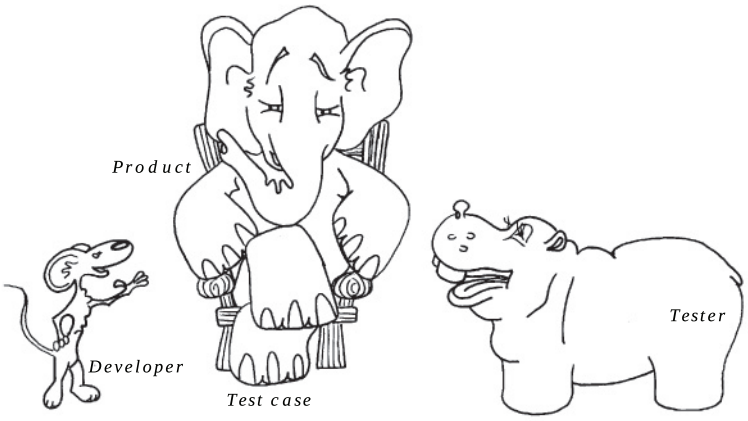
Tests are easily integrated and can be run automatically for regression testing.

(f) Name one disadvantage of using JUnit for testing

JUnit can be used only for unit testing (and not integration testing, configuration, etc.)

6 points

Q 6 What are the roles of the three animal characters shown on the cover page of "Foundations of Software Testing" (and reproduced below) used in CS 490M ? Express your answer using terminology from software engineering.



One large pan pizza with 3 toppings of your choice to be shared by all those that get the correct answer.

None of the given answers was correct though two were close. Continue thinking !

End of Questions.