CS510 Midterm Solutions (2012 Spring)

April 5, 2012

Name: ____________________________
1 Testing (25p)

(a) (Combinatorial Testing) (15p)

Assume a program has three factors: A, B, and C. The levels of these factors are \( \{a_1, a_2\} \), \( \{b_1, b_2, b_3\} \), and \( \{c_1, c_2\} \).

Compute the pair wise cover array using the IPO algorithm.

**Answer:**

Consider parameters in order A, B, C:

\[ a_1b_1c_1 \]
\[ a_1b_2c_2 \]
\[ a_1b_3c_1 \]
\[ a_2b_1c_2 \]
\[ a_2b_2c_1 \]
\[ a_2b_3c_2 \]

(b) (Mutation testing) (10p)

1. input (i);
2. if (i<10) {
3.   if (i>5)
4.     print ("5<i<10");
5.   else
6.     print ("i<=5");
7 } else
8. print ("i>=10");

Assume we have three mutants. One is "i<10" at line 2 is mutated to "i<=10", the second is that "i<10" is mutated to "i>10", and the third is that "i>5" is mutated to "i>=5".

Assume the test suite is \{i=6, i=10\} and the oracle is purely based on the program output. What is the mutation coverage?

**Answer:**
\begin{tabular}{|c|c|c|}
\hline
Mutant & $i = 6$ & $i = 10$ \\
\hline
1 & $i < 10 \Rightarrow i \leq 10$ & \checkmark (pass) & \xmark (fail) \\
2 & $i < 10 \Rightarrow i > 10$ & & \\
3 & $i > 5 \Rightarrow i \geq 5$ & \checkmark & \checkmark \\
\hline
\end{tabular}

Mutation coverage = 2/3

\section{Statistical Debugging (25p)}
Assume the following program and eight executions, including both passing and failing.

(a) Please compute the suspiciousness of the statements based on the Tarantula algorithm (15p).

**Answer:**

\[ F(s) \text{ and } P(s): \text{Number of failing and passing runs that execute } s \]
\[ |P| \text{ and } |F|: \text{Total number of passing and failing runs} \]

\[ Suspiciousness(s) = \frac{F(s)}{|P|} + \frac{P(s)}{|F|} \]

\[ |F| = 2 \]
\[ |P| = 6 \]

\begin{tabular}{|c|c|}
\hline
s & Suspiciousness(s) \\
\hline
1 & 1/2 \\
2 & 1/2 \\
3 & 1/2 \\
4 & 3/4 \\
5 & 1/2 \\
6 & 3/5 \\
7 & 0 \\
\hline
\end{tabular}

(b) Assume the two predicates at lines 3 and 5 are monitored. Please compute the suspiciousness of them according to the Scalable Remote Bug Isolation algorithm (10p).

Please briefly present the formula you use in case you miscalculate.

1. x=1; 

2. i=input();
3. if (i%2==0)
4. x=x+i/2;
5. if (x%2==1)
6. print ("Odd.");
7. else print ("Even.");

<table>
<thead>
<tr>
<th>i=</th>
<th>Statement</th>
<th>Output</th>
<th>Passing/ Failing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>* * * *</td>
<td>Odd</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>* * * * *</td>
<td>Even</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>* * * *</td>
<td>Odd</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>* * * * *</td>
<td>Odd</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>* * * *</td>
<td>Odd</td>
<td>P</td>
</tr>
<tr>
<td>6</td>
<td>* * * * *</td>
<td>Even</td>
<td>P</td>
</tr>
<tr>
<td>7</td>
<td>* * * *</td>
<td>Odd</td>
<td>P</td>
</tr>
<tr>
<td>8</td>
<td>* * * * *</td>
<td>Odd</td>
<td>F</td>
</tr>
</tbody>
</table>

Answer:

\[
\text{failure}(p) = \frac{F(p)}{F(p) + P(p)}
\]

\[
\text{context}(p) = \frac{F'(p)}{F'(p) + P'(p)}
\]

\[
\text{Suspiciousness}(p) = \text{failure}(p) - \text{context}(p)
\]

\[
\text{Suspiciousness}(i%2 == 0) = 1/4
\]

\[
\text{Suspiciousness}(x%2 == 1) = 1/12
\]
3  CFG and Path Encoding (30p)

1. input(a,b,c);
2. z=0;
3. while (a>0) {
4.   if (a%b!=0) {
5.     c=c-1;
6.     if (c>a)
7.       z=z+1;
8.     else
9.       break;
10.   }
11.   a--;
12. }
13. print z;

Please present the CFG (10p) and the path encoding graph of the above program (10p). List the encodings for individual paths (10p).

Answer:
Counter[r]++
R=4
Counter[r]++
R+=3
R+=2
R+=1

Figure 1: Control Flow Graph
Figure 2: Path Encoding Graph
4 Slicing, 16 points

(a) What is the static slice of z at 13 (5p)?

Answer:
\{1, 2, 3, 4, 5, 6, 7, 11, 13\}

(b) What is the static slice of a at 11 (5p)?

Answer:
\{1, 3, 4, 5, 6, 11\}

(c) Leverage the program in Problem 3 to explain the differences (at least one aspect) between static and dynamic slicing (6p). You may want to use an execution and its corresponding dynamic slice to illustrate the comparison.

Answer:
Dynamic slicing only includes the executed statements that actually contributed to the value. Consider \(a = -1\), then the while loop does not execute. So, the dynamic slice of \(z@13\) is \{2, 13\}. 
5 Misc. (4p)

Sketch a dynamic analysis that can detect heap buffer overflows.

**Answer:**

Use shadow memory to identify allocated heap from unallocated heap. For heap addresses inside `malloc` region set $SM[addr] = 1$ and for the others set $SM[addr] = 0$.

Shadow memory will be updated when heap is allocated and released when a read/write is performed, check whether the address is in allocated memory (i.e. $SM[addr] = 1$).