Term Test 2

CSE 4313 3.0 Software Engineering Testing

Section M, Winter 2008

Family Name: _____

Given Name(s): _____

Student Number: |___ |___ |___ |___ |___ |___ |___ |___ |___ |

Question	Out of	Mark
Q1	24	
Q2	12	
Q3	14	
Q4	30	
Q5	20	
Total	100	
Letter grade		

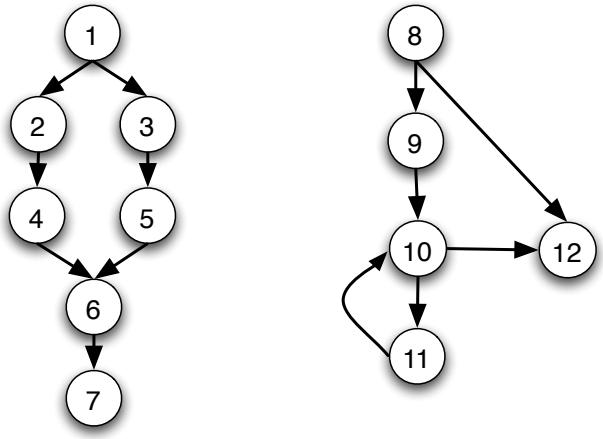
1. **[24 marks]** Consider the following two methods

```
1   int m1(int a, int b) {
2      int result = 0;
3      if (a>0) {
4          result ++;
5          result = m2(result, -4);
6      }
7      else {
8          result = m2(result, 7);
9          result --;
10     }
11     result = result * b;
12     result = m2(result, b);
13     return result;
14  }
15
16  int m2 (int r, int b) {
17      if (b > 0) {
18          for (int i=0; i<5; i++) r = r + b;
19      }
20      return r;
21  }
```

Assume that execution starts with `m1`. In the next two pages, do the following:

- Draw the MM-path graph for this system. For each method, clearly identify the module execution paths.
- Identify a set of test cases that will result in complete path coverage of the MM-path graph (you need only consider feasible MM-paths). You only need to provide the values for arguments `a` and `b`, i.e. you do not have to compute the value of `result`.

The CFGs for the two methods are shown below



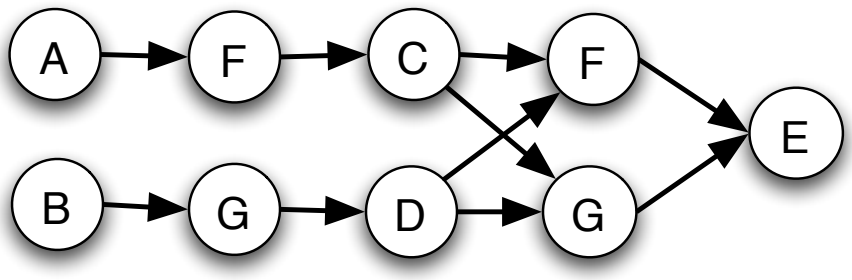
Method `m1` contains 5 module execution paths:

A: 1-2
B: 1-4
C: 3-6
D: 5-6
E: 7

Method `m2` contains 2 module execution paths:

F: 8-12
G: 8-9-(10-11)*-12

The MM-path graph is shown below (only feasible paths are shown)



There are 4 paths is the MM-path graph. As a result, 4 test cases will be needed for path coverage. Careful inspection indicates that test cases that fulfill the following constraints are needed:

$a > 0, b > 0$
$a > 0, b \leq 0$
$a \leq 0, b > 0$
$a \leq 0, b \leq 0$

Marking scheme: 7 for the module execution paths, 9 for the MM-path graph (-2 per error), 8 for the four test cases.

2. **[12 marks]** Consider the following method that sums up the elements of an array of integers:

```
1  public static int sum(int[] x)
2  {
3    int s = 0;
4    for (int i=0; i < x.length; i++)
5    {
6      s = s + x[i];
7      // s = s - x[i];
8    }
9    return s;
10 }
```

Line 7 is a mutation to the original program (in the mutant program it is uncommented, while line 6 is commented out). In the following, identify, and explain briefly how you derived, a test case that:

(a) Does not reach the mutant.
An empty array [3 marks]

(b) Reaches the mutant, but does not infect the program.
An array containing only 0s [3 marks]

(c) Infects the program, but does not propagate to the output.
An array whose elements add up to 0 [3 marks]

(d) Kills the mutant.
Any other array [3 marks]

3. **[14 marks]** Consider a logic function Z of three boolean variables A, B, C. The formula for Z is

$$Z = A+ \sim B \sim C$$

Describe the process of applying the Variable Negation test strategy. What is the minimum test suite suggested by this strategy?

Following is the truth table (not necessary to draw for full marks).

Variant	A	B	C	Z
0	0	0	0	1
1	0	0	0	0
2	0	0	1	0
3	0	0	1	0
4	0	1	0	1
5	0	1	0	1
6	0	1	1	1
7	0	1	1	1

The following table presents all the candidate sets (each worth 2 marks).

Candidate set #	Term	Type	Set
1	`A`	Unique true point	{5,6,7}
2	`A`	Near false point `~A`	{1,2,3}
3	`~B~C`	Unique true point	{0}
4	`~B~C`	Near false point `~BC`	{1}
5	`~B~C`	Near false point `B~C`	{2}

Candidate sets of cardinality 1 determine that variants 0,1,2 must be part of the test suite. This covers all candidate sets except for the first one. We select one member from that set as well. The final test suite (worth 4 marks) will contain the following test cases:

0
1
2
5 or 6 or 7

4. **[30 marks]** Consider the following method that implements the Triangle classification problem:

```
1   String classify(int a, int b, int c) {
2       if (!((a + b > c) && (b + c > a) && (a + c > b)))
3           return "Not a triangle";
4
5       int match = 0;
6
7       if (a == b) match = match + 1;
8       if (b == c) match = match + 1;
9       if (a == c) match = match + 1;
10
11      if (match == 0) return "Scalene";
12      if (match == 1) return "Isosceles";
13      else return "Equilateral";
14  }
```

(a) With respect only to variable `match`, compute the value of the All-Defs and the All-Uses coverage criteria for a test suite that contains only the test case (3,3,3) [20 marks].

(b) What other test cases would you add in order to maximize the value of the All-Uses coverage criterion and why? [10 marks]

Variable `match` has four definitions in lines 5,7,8,9. It also has five uses in lines 7,8,9,11,12.

When all three inputs are equal, then all definitions are reached and followed to a use. Therefore, the value of the All-Defs criterion is 100%. [10 marks]

Of the 20 possible d-u paths, only the following 5 are exercised by the (3,3,3) test case: 5-7 ,7-8, 8-9, 9-11, 9-12. As a result, the value of the All-Uses criterion is 5 out of 20, or 25%. [10 marks]

Not surprisingly, test cases for isosceles triangles (all three possibilities) as well as one for a scalene triangle will increase the value of the All-Uses criterion as shown below

$b = c \neq a$ covers path 5-8
$a = c \neq b$ covers path 5-9
$a = b \neq c$ covers paths 7-11 and 7-12
$b \neq c \neq a$ covers paths 5-11 and 5-12

The remaining paths are not feasible.

5. **[20 marks]** The Unix utility `wc` receives a set of filenames as input, and outputs information about the number of lines, the number of words, and the number of characters in each file as well as in total. Lines are separated by NEWLINE characters, while words are separated by SPACE, TAB, or NEWLINE characters. Here is a typical output:

```
& wc test2.tex test2.pdf
     195     677    4573 test2.tex
     593    2498   34017 test2.pdf
     788    3175   38590 total
```

Your task is to describe the application of any software testing strategies that you believe are appropriate in order to test this application. The derived test cases must be identified clearly in your answer. Answers that list test cases without describing how they were derived will receive a poor mark.

The specification is provided as is. You may have to make assumptions about system behaviour that is not mentioned in the specification. If this is the case, choose reasonable ones and state them explicitly.

This question can be answered in a number of different ways involving Boundary Value Analysis and Equivalence Class Testing. Any of the following sets of tests cases was worth 2 marks:

- Non-existant files
- Filenames including full paths
- Files that do not have the right permissions
- Number of files: 0, 1, large
- Lines in the file; 0, 1, large
- Words in the file; 0, 1, large
- Characters in the file; 0, 1, large
- Number of words per line: 0, 1, large
- Number of characters per word: 0, 1, large
- Word separator type
- Separator length: 1, more

Other interesting sets of test cases were also given 2 marks.