MIDTERM W2015

Student ID: **Student Name:**

Signature: **Course Number:**

©2015 University of Waterloo
Department of Electrical and Computer Engineering
SE465/ECE453/CS447/ECE653/CS647 Software Testing, Quality Assurance & Maintenance
Instructors: Lin Tan & Patrick Lam
Examination Date and Time: Monday March 2, 2015, 7:00PM - 8:20PM

Rooms:	ECE 453	MC-4042	
	ECE 653	MC-4041	
	CS 647	MC-4041	
	CS 447	MC-4042	**20446079–20595284**
	CS 447	MC-4060	**20395561–20444987**
	CS 447	MC-4063	**20309119–20393653**
	SE 465 - 001	MC-4040	**20333438–20432077**
		MC-4058	**20433910–20460755**
		QNC-1506	**20460831–20465276**
		EV3-3412	**20465562–20472536**
		EV3-4412	**20473079–20592531**
	SE 465 - 002	RCH-110	**20310541–20381100**
		DWE-3518	**20381206–20390487**
		DWE-3522A	**20391355–20598426**

Instructions:

- You have **80 minutes** to complete the exam.
- You can bring **printed or handwritten** material, e.g., books, slides, notes, etc.
- If you separate the pages, make sure your names and student IDs are on the top of every page.
- Unauthorized duplication or archival of these questions is not permitted. To be returned with the exam booklet after the completion of the exam.
- If information appears to be missing from a question, make a reasonable assumption, state your assumption, and proceed. Do not simplify the question.
- Attempt to answer questions in the space provided. If necessary, you may use the back of another page. If you do this, please indicate it clearly.
- **Illegible answers receive NO point.**

Question	Mark	Points
Q1		**20**
Q2		**15**
Q3		**15**
Q4		**15**
Q5		**35**
Total:		**100**

Question 1 (20 points)

(a) **Faults and failures.** Define the following two terms: fault and failure. Give one example of a fault and one failure scenario using a valid C/C++ or Java code snippet and any relevant inputs and outputs. [5 points]

(b) **Subsumption.** All-Defs Coverage does not subsume All-Uses Coverage. Give an example showing why not. You must provide a valid C/C++ or Java code snippet. [5 points]

(c) **Tools.** One of these tools is not like the others. Which one, and why? Coverity Static Analyzer, FindBugs, GrammaTech CodeSonar, and Valgrind. [5 points]

(d) **Concurrency bugs.** Point out the fault and propose a fix. [5 points]

```
1  /* 2.4.0:drivers/sound/cmpci.c:cm_midi_release: */
2  lock_kernel();
3  if (file->f_mode & FMODE_WRITE) {
4    add_wait_queue(&s->midi.owait, &wait);
5    ...
6    if (file->f_flags & O_NONBLOCK) {
7      remove_wait_queue(&s->midi.owait, &wait);
8      set_current_state(TASK_RUNNING);
9      return -EBUSY;
10   }
11   ...
12 }
13 unlock_kernel();
```

Question 2 (15 points)

Function inspired from http://stackoverflow.com/questions/13122696/drawing-a-control-flow-graph

```
1  double foo(int x, int y, int power) {
2   if (y<0)
3           power=-y;
4   else
5           power=y;
6   double z=1;
7   while (power != 0) {
8           z=z*x;
9           power=power-1;
10  }
11  if (y<0) {
12     z=1/z;
13  }
14  return z;
15 }
```

(a) Draw a control flow graph (CFG) for function `foo` (9 basic blocks). You must use only the line numbers provided above as the content of each node (e.g., 20–25, if lines 20–25 belong to a basic block), and label the nodes using only circled lowercase letters (ie nodes (a), (b), (c), (d), (e), (f), (g), (h), and (i)). Ensure the graph is clear and legible. Unclear graphs will not receive full points. [5 points]

(b) List the set of test requirements for Edge Coverage. Give a set of test cases satisfying Edge Coverage. [5 points]

(c) Does your test set cover the bug on line 12 (potential division by 0)? If not, provide an input triggering the bug which is not in your test set. If yes, provide a test set satisfying Edge Coverage which does not trigger the bug. [5 points]

Question 3 (15 points)

Propose two distinct non-stillborn and non-equivalent mutants for the following method. Say which mutation operator you used and where you apply it. Show that you can kill the mutants, demonstrating that they are non-equivalent mutants, by writing down test cases and relevant outputs for each of these mutants and the original method.

```
1  public static int odd(int[] x)  {
2  // Effects: if x==null throw NullPointerException,
3  // else return the number of elements in x that are odd
4            int count = 0;
5            for (int i =0; i < x.length; i++) {
6                   if (x[i]%2==1 || x[i]%2==-1 ) {
7                     count++;                                  }
8            }
9            return count;
10 }
```

Question 4 (15 points)

Program slicing is a technique for automatically simplifying programs by omitting irrelevant operations. Slicing reduces a program to a minimal form which preserves the behaviour of a selected statement. Program slicing can be used in debugging to locate source of errors more easily by ruling out statements that do not contribute to a faulty behaviour. In this question, you will apply program slicing to analyze a real bug in Mozilla.

An example of program slicing is as follows:

```
1  int i;
2  int sum = 0;
3  int product = 1;
4  for(i = 1; i < N; ++i) {
5    sum = sum + i;
6    product = product * i;
7  }
8  write(sum);
9  write(product);
```

We can slice this program with respect to the statement `(write(sum))`, thus omitting, for instance, all calculations of the irrelevant `product` variable. A slice of a program with respect to a statement s must include all statements which (transitively) define values that are used in s (that is, all statements that contribute data flow to s). Furthermore, a slice must also include all control-flow statements s' which affect whether or not s is reachable, along with all statements that contribute data flow to s'.

For `write(sum)`, we keep the def of the `sum` variable in the loop; that def also uses the initial def of `sum` before the loop.

```
1  int i;
2  int sum = 0;
3
4  for(i = 1; i < N; ++i) {
5    sum = sum + i;
6
7  }
8  write(sum);
```

The Mozilla bug reported in its bug report 192226 was caused by the code below. The conditional expression in the second if block (Line 1923) leads to a Null Pointer Exception at "`child = child.getNext().getNext();`" (Line 1927).

```
1876 private void visitRegularCall(Node node, int type,
                                  Node child, boolean firstArgDone)
1878{
                ...
1895    int childCount = 0;
1896    int argSkipCount = (type == TokenStream.NEW) ? 1 : 2;
1897    while (firstArgDone && (child != null)) {
1898       childCount++;
1899       child = child.getNext();
1900    }
                ...
1905    int argIndex = -argSkipCount;
1906    if (firstArgDone && (child != null)) {
1907        child = child.getNext();
1908        argIndex++;
1909        aload(contextLocal);
1910        addByteCode(ByteCode.SWAP);
1911    }
                ...
1918    boolean isSpecialCall = node.getProp(Node.SPECIALCALL_PROP) != null;
1919    boolean isSimpleCall = false;
1920    String simpleCallName = null;
                ...
1922    simpleCallName = getSimpleCallName(node);
1923    if (simpleCallName != null && !isSpecialCall) {
1924        isSimpleCall = true;
1925        push(simpleCallName);
1926        aload(variableObjectLocal);
1927        child = child.getNext().getNext();
1928        argIndex = 0;
1929        push(childCount - argSkipCount);
1930        addByteCode(ByteCode.ANEWARRAY, "java/lang/Object");
1931    }

        ...
2053}
```

(a) Give the program slicing of method `visitRegularCall(...)`, with respect to the statement `(child = child.getNext().getNext())`. You may indicate the slice by writing down a sequence of line numbers, or by copying the lines themselves along with the line numbers. [10 points]

(b) On the program slice generated in part (a), identify one du-path with respect to `child` by providing the sequence of line numbers in the path. You don't need to draw the full Control Flow Graph. [5 points]

Question 5 (35 points)

The following pair of recursive functions, together, returns the total number of leaf nodes in a Tree data structure. The user calls `countLeaves()` on the root node of a Tree to begin recursion (`CountLeaves(myTree.rootNode)`).

- `Node.numChildren()` returns the number of child nodes of a `Node`.
- `Node.getChildren()` returns a `NodeList` object consisting of a list of the child nodes of the current node, or null if no child nodes exist.
- `NodeList.count()` returns the number of nodes in the `NodeList`.
- `NodeList.getNode(int i)` returns the `i`'th `Node` in the `NodeList`.

```
1  int countLeaves(Node m) {
2    int c = m.numChildren();
3    int total = 1;
4    if (c != 0) {
5      NodeList children = m.getChildren();
6      total = countLeavesInNodeList(children);
7    }
8    return total;
9  }
```

```
1  int countLeavesInNodeList(NodeList nl) {
2    int total = 0;
3    if (nl == null) {
4      return total;
5    }
6    for (int i = 0;
7             i < nl.count();
8             i++) {
9      Node n = nl.getNode(i);
10     total += countLeaves(n);
11   }
12   return total;
13 }
```

(a) Draw the minimal node (hint: 6) control flow graph (CFG) for the function `countLeavesInNodeList()`. You must use only the line numbers provided above as the content of each node (e.g., 20–25, if lines 20–25 belong to a basic block), and label the nodes using only circled lowercase letters (ie nodes (a), (b), (c), (d), (e), and (f)). Ensure the graph is clear and legible. Unclear graphs will not receive full points. [6 points]

(b) Create the Def-Use CFG for variables `nl`, `i`, and `total` in `countLeavesInNodeList()` by reproducing the graph from a), filling the nodes with def-set(s) and use-set(s). Include the implicit definition of `nl` at the beginning of the function as part of (a). Ensure the graph is clear & legible. Unclear graphs will not receive full points. [6 points]

(c) List test requirements for Prime Path Coverage (PPC) for `countLeavesInNodeList()`, including cycles. [7 points]

(d) Enumerate def-pair sets and list resulting test requirements for All-Uses Coverage (AUC) for variables `nl` and `i` in `countLeavesInNodeList()`. Note: `nl.getNode(i)` is a use of `nl`, `nl.count()` is a use of `nl`. [7 points]

(e) List three inter-procedural DU pairs related to `countLeavesInNodeList()`. Use a tuple (function name, variable name, line number) to denote a location. For example,

```
(countLeaves, total, 6) -> (countLeavesInNodeList, total, 10)
```

denotes an inter-procedural DU pair: the last def of `total` in line 6 of method `countLeaves` reaches the first use of `total` in line 10 of method `countLeavesInNodeList`. (Don't use that example in your answer). [9 points]

(This page is intentionally left blank.)