Homework 3: Symbolic Execution and Concolic Testing

General Directions

This homework is intended to be an individual one, however, you can work together in a group as long as you follow the policy on teamwork found in the course grading policies webpage.

What to turn in

For this homework, turn in a PDF, Microsoft Word document, or text file on Webcourses.

The Assignment

1. [Plan] [Architect] Consider again an online banking system, which would be accessible from a computer or smart phone owned by a customer. The bank's software developers need your direction as to what tools would help them most avoid the most important threats that they face.
   (a) (6 points) To protect the bank against the most important threat (as described in your answer to the next part), it is important to identify what that threat is. What is the most important threat to the bank's online banking system (that your next answer will help protect against)? Briefly name or describe it and briefly say why it is important.
   (b) (5 points) Assume that the developers at the bank all have at least a bachelor's degree in Computer Science, and that most have at least a few months of experience with software development. Given this background, which kind of software analysis tool would be best for protecting against the most important threat (which you identified in the previous answer)? Pick one of the following as your answer: static program analysis, symbolic execution, concolic testing, or fuzz testing.
   (c) (10 points) Justify your answer to the previous part; that is, briefly say why the kind of software analysis tool (that you answered in the previous question) would be best to protect against the most important threat to the bank, compared to the other choices (offered in the previous question).

2. Consider the following Java method (which is part of a file `HW3.java` available with this homework).

```
public int check(int i, int j, int k) {
  int x = 0; // l4
  int y = 0; // l5
  if (i>0) { //l6
    l7: y = 11;
  } else {   // l8
    l9: y = 9;
  }
  ; // l11
  if (j>7) { // l12
    l13: x = 6;
    if (i <= 0 && k == 0) { // l14
      l15: y = y-1;
    }
  }
  ; // l18
  l19: assert x+y != 9;
  return x+y;
}
```

(a) (10 points) Is the assertion at the end (at label `l19`) always true (i.e., in all possible executions)? If yes, briefly explain why; if no, give an example of 3 concrete inputs (for `i`, `j`, and `k`) that makes the assertion fail.

(b) (5 points) Suppose the program you were to investigate with contained over a million lines of code, would it be better to use to use a tool or a manual code review to check for vulnerabilities? Briefly justify your answer.

(c) (5 points) Suppose we wanted to find vulnerabilities or errors (or prove that they do not exist) in the least amount of time (combined human and computer) time. In that case, what kind of tool would be best to review the code automatically? That is, if the assertion can never fail, what kind of tool would most efficiently prove that, or if the assertion can fail, what kind of tool would help you find an example that demonstrates the failure most efficiently?

3. Download and install the kind of tool you named in your answer to the last question above, and document:

(a) (5 points; extra credit) how difficult it was to install on your machine,

(b) (5 points; extra credit) what kind of computer and operating system you are using,

(c) (5 points; extra credit) how long it took that tool to either prove that the assertion would never be violated or to find an example of how the assertion could be violated, and

(d) (5 points; extra credit) the output of the tool.