SENG3320/6320 Software Verification and Validation
Semester 1, 2020

Assignment 2 – Automated Test Data Generation

25 marks (25% of the whole course assessment)
Due 11:59pm, Monday, 8 June 2020

1. Fuzz Testing (12.5%)

Fuzz Testing is a random testing technique, which is commonly used to discover software crashes through a large amount of random data. It is a cost-effective alternative to more systematic testing techniques.

In this assignment, you are to apply fuzz testing to test a KWIC program. The KWIC (Key Word In Context) problem is defined as follows:

"The KWIC [Key Word in Context] index system accepts an ordered set of lines; each line is an ordered set of words, and each word is an ordered set of characters. Any line may be "circularly shifted" by repeatedly removing the first word and appending it at the end of the line. The KWIC index system outputs a list of all circular shifts of all lines in alphabetical order."

The KWIC system is a real system and is widely used in keyword in context indexes for libraries.

Example

Consider the following three book titles (lines):
– Pattern-Oriented Software Architecture
– Software Architecture
– Introducing Design Patterns

The KWIC system produces the following output:
– Architecture Software
– Architecture Pattern-Oriented Software
– Design Patterns Introducing
– Introducing Design Patterns
– Patterns Introducing Design
– Pattern-Oriented Software Architecture
– Software Architecture Pattern-Oriented
– Software Architecture

Users can now quickly search for book titles that contain phrases such as "Software Architecture" or "Design Pattern".

For this assignment, you are given a Java implementation of the KWIC problem (KWIC.class), which can be found in Blackboard. The usage of this program is:

java KWIC input.txt

where input.txt is a plain text file containing the input book titles (each line is a title).

You are required to apply fuzz testing to generate test input to crash the given KWIC program. More specifically, you are required to:

- Perform manual black-box testing (manual test case design based on the problem specification)

- Performance testing: measure the runtime performance (in terms of execution time) of the KWIC program on the given books.txt file. Please change the size of the file (by adding or deleting some book titles) to measure the runtime performance with different file sizes.
- Develop a fuzz testing tool, which can feed a large amount of random data (random book titles such as “a8h h19%p”) to the KWIC program in an attempt to make it crash.
- Perform the fuzz testing, record the number of unique crashes (i.e., crashes with different exception messages at different program locations/line numbers) and the test input that crashes the program.
- Write a test report, which describes the test tool design, test environment, example of test cases, and summary of test results (especially the number of unique crashes detected).

The marks will be distributed as follows:

Manual black-box testing of the KWIC program	2 marks
Performance testing	1 mark
Fuzz tool development	3 marks
Fuzzing testing of the KWIC program	4 marks
Test report	2.5 marks

2. Symbolic Execution (12.5%)

As described in the lecture, symbolic execution is a means of analyzing a program to determine what inputs cause each part of a program to execute. It ‘executes’ programs with symbols rather than concrete input. In this assignment, you are to experiment with symbolic execution based test data generation using KLEE, a symbolic execution tool for C programs. More specifically, you will:

- Download and install the KLEE tool from http://klee.github.io/
- Apply KLEE to generate test data for the Triangle program as shown below.
- Compute the control-flow coverage of the test data generated by KLEE, including statement, branch decision coverage, condition coverage, condition/decision coverage, and multiple condition coverage.
- Apply fuzz testing to generate test data for the Triangle program, compare the test results of fuzz testing and symbolic execution (in terms of the control-flow coverage achieved and the time spent).
- Write a test report for this experiment. The test report should describe your experimental design, steps, and results.

```
/* The Triangle program, which determines if three inputs specify an equilateral triangle, an isosceles
triangle, an ordinary triangle, or non-triangle. */
void triangle (int a, int b, int c){
if ((a+b>c)&&(a+c>b)&&(b+c>a)) {
   if (a==b || a==c || b==c) {
      if (a==c && a==b)
         printf("equilateral triangle .\n");
      else if (a==c||b==c)
         printf("isosceles triangle.\n");
   }
   else
       printf("triangle.\n");
}
else
   printf("non-triangle.\n");

return;
}
```

The marks will be distributed as follows:

Successful application of KLEE to the testing of the Triangle program	4 marks
Control flow coverage analysis	3 marks
Fuzz testing	3 marks
Test report	2.5 marks

This assignment should be submitted via Blackboard. The submission should be a single zip file, which contains all source code, program under test, test data, test report, and supporting materials. Copying, plagiarism and other malpractices are not allowed. Write in your own words.

Please remember to attach an Assignment Cover Sheet to each submission. Please also state clearly the contribution of each group member to this assignment.

Start early and work efficiently and effectively with your group members!

- End -