SENG3320/6320 Software Verification and Validation
Semester 1, 2022

Assignment 2 – Automated Test Data Generation

100 marks (25% of the whole course assessment)
Due 11:59pm, Friday, 3 June 2022

1. Fuzz Testing (40%)

Fuzz Testing (random testing) is commonly used to discover software crashes through a large amount of randomly generated 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:

1) 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.
2) 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.

3) 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:

Fuzz tool development	10 marks
Fuzzing testing of the KWIC program	20 marks
Test report	10 marks

2. Automated Testing Techniques (60%)

In this assignment, you are to experiment with symbolic execution, mutation testing, and fuzz testing techniques. For symbolic execution, you are to experiment with KLEE, a symbolic execution tool for C programs. More specifically, you will:

1) Symbolic execution: a) Download and install the KLEE tool from http://klee.github.io/; b) Apply KLEE to generate test data for the Triangle program as shown below.
2) 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.
3) Fuzz testing: 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).
4) Mutation testing: apply at least two mutation operators to the Triangle program, and perform mutation testing. a) Mutate the Triangle code using the two mutation operators you selected. At least 10 mutants should be generated. The mutation can be done manually (by hand) or automatically (by developing a simple mutation tool); b) Perform mutation testing with the test cases generated by symbolic execution; and c) Measure the number of mutants being killed by the test cases. If some mutants are not killed, add more test cases to kill the mutants.
5) Write a test report for this experiment. The test report should describe your experimental design, steps, and results. It should also include a comparison of symbolic execution, mutation testing, and fuzz testing techniques used in this experiment.

```
/* 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	15 marks
Control flow coverage analysis	10 marks
Fuzz testing	10 marks
Mutation testing	15 marks
Test report	10 marks

This assignment should be submitted via Canvas. 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 -