# SENG3320 Assignment 2: Automated Test Data Generation

|  |  |
| --- | --- |
| **Group 10** | |
| Kyle Beattie | C3303374 |
| Ni Zeng | C3238805 |
| Brandon Allen | C3279505 |
| Austin Baxter | C3356468 |

**Table of Contents**

[SENG3320 Assignment 2: Automated Test Data Generation 1](#_Toc103869436)

[1. Question 1: Fuzz Testing 2](#_Toc103869437)

[1.1. Test Tool Design 2](#_Toc103869438)

[1.2. Test Environment 2](#_Toc103869439)

[1.3. Test Cases 2](#_Toc103869440)

[1.4. Summary 2](#_Toc103869441)

[2. Question 2: Automated Testing Techniques 2](#_Toc103869442)

[2.1. Symbolic Execution 2](#_Toc103869443)

[2.2. Fuzz Testing 2](#_Toc103869444)

[2.3. Mutation Testing 2](#_Toc103869445)

[2.4. Comparison 2](#_Toc103869446)

## Question 1: Fuzz Testing

### Test Tool Design

### Test Environment

### Test Cases

### Summary

## Question 2: Automated Testing Techniques

### Symbolic Execution

### Fuzz Testing

The idea of Fuzz Testing on this question is to apply random integers for variables a, b, and c to examine the outcome of the triangle (int a, int b, int c).  
  
Fuzz Testing structure:  
>FuzzTesting (Q2 Fuzz Testing task folder)

FuzzInput\_Output.txt (contain the input & output test case result)

FuzzTesting.c (Fuzz test case generator)

FuzzTesting.exe (Fuzz test case generator executor)

triangle. c (Given c program for Fuzz Testing)  
  
When run the FuzzTesting.exe, the Fuzz generator will first ask for user input for the number of test cases that need to be generated. After the input, the generator will generate 3 random integer numbers (a,b,c) from the range 0 to 9 for each of the test cases. The generated integer number(a,b,c) will execute using triangle.c and record the input and output result in a text file Name ‘Fuzzinput\_Output.txt’.  
  
In the text file “Fuzzinput\_Output.txt”, the text file will first list individual values a,b,c and results of each test case. The total number of test cases, total time spent fuzz testing, number of non-triangles, triangles, isosceles triangles and equilateral triangles and number of errors that occurred.  
  
In terms of control-flow coverage covered by Fuzz Testing is very depends on the number of test cases to be generated. The more test cases to be generated, the more coverage will be covered.  
During the Fuzz Testing on triangle.c , it discovered that there are some test cases with no output type of triangle , it is certainly a bug in trianle.c. the total number of errors has been recorded in the “Fuzzinput\_Output.txt” File.  
The main problem that causes this no out for a test case is that in triangle.c , there is one more condition which is not been added to the code.  
the missing condition: (a == b )   
the original code in triangle.c :   
Text

Description automatically generated  
it should be :  
 else if (a ==c || b== c || a ==b) printf(“isosceles triangle.\n”);  
  
Compare the test results of Fuzz testing and symbolic execution:

|  |  |
| --- | --- |
| **Control-flow coverage achieved:** | |
| Fuzz Testing: | Symbolic testing: |
| **Time spent:** | |
| Fuzz Testing: | Symbolic testing: |

### Mutation Testing

### Comparison