COSC 3P95- Software Analysis & Testing

Assignment 1

Due date: Monday, Oct 16th, 2023, at **23:59** (11:59 pm)

Delivery method: This is an individual assignment. Each student should submit one PDF through Brightspace.

Attention: This assignment is worth 10% of the course grade. Please also check the Late Assignment Policy.

Name: _____   **Student ID:** _____

Questions:

1- Explain the difference between "sound" and "complete" analysis in software analysis. Then, define what true positive, true negative, false positive, and false negative mean. How would these terms change if the goal of the analysis changes, particularly when "positive" means finding a bug, and then when "positive" means not finding a bug. **(10 pts)**

2- Using your preferred programming language, implement a random test case generator for a sorting algorithm program that sorts integers in ascending order. The test case generator should be designed to produce arrays of integers with random lengths, and values for each sorting method.
   A) Your submission should consist of:
   a. Source code files for the sorting algorithm and the random test case generator.
   b. Explanation of how your method/approach works and a discussion of the results (for example, if and how the method was able to generate or find any bugs, etc.). You can also include bugs in your code and show your method is able to find the input values causing that.
   c. Comments within the code for better understanding of the code.
   d. Instructions for compiling and running your code.
   e. Logs generated by the print statements, capturing both input array, output arrays for each run of the program.
   f. Logs for the random test executions, showing if the test was a pass and the output of the execution (e.g., exception, bug message, etc.).

   B) Provide a context-free grammar to generate all the possible test-cases. **(18 + 8 = 26 pts)**

3- A) For the following code, manually draw a control flow graph to represent its logic and structure.

```python
def filterData(data, limit, exceptions):
    filtered_data = []
    index = 0
    while index < len(data):
        item = data[index]
        if item in exceptions:
            modified_item = item + "_EXCEPTION
        elif item > limit:
            modified_item = item * 2
        else:
            modified_item = item / limit

        filtered_data.append(modified_item)
        index += 1

    return filtered_data
```

The code is supposed to perform the followings:
   a. If an item is in the exceptions list, the function appends "_EXCEPTION" to the item.
   b. If an item is greater than a given limit, the function doubles the item.
   c. Otherwise, the function divides the item by 2.

B) Explain and provide detailed steps for "random testing" the above code. No need to run any code, just present the coding strategy or describe your testing method in detail. **(8 + 8 = 16 pts)**

4- A) Develop 4 distinct test cases to test the above code, with code coverage ranging from 30% to 100%. For each test-case calculate and mention its code coverage.

B) Generate 6 modified (mutated) versions of the above code.

C) Assess the effectiveness of the test cases from part A by using mutation analysis in conjunction with the mutated codes from part B. Rank the test-cases and explain your answer.

D) Discuss how you would use path, branch, and statement static analysis to evaluate/analyse the above code. **(4 * 8 = 32 pts)**

5- The code snippet below aims to switch uppercase characters to their lowercase counterparts and vice versa. Numeric characters are supposed to remain unchanged. The function contains at least one known bug that results in incorrect output for specific inputs.

```
def processString(input_str):
    output_str = ""
    for char in input_str:
        if char.isupper():
            output_str += char.lower()
        elif char.isnumeric():
            output_str += char * 2
        else:
            output_str += char.upper()

    return output_str
```

In this assignment, your tasks are:
   a. Identify the bug(s) in the code. You can either manually review the code (a form of static analysis) or run it with diverse input values (a form of manual random testing). If you are unable to pinpoint the bug using these methods, you may utilize a random testing tool or implement random test case generator in code. Provide a detailed explanation of the bug, identify the line of code causing it, and describe your strategy for finding it.
   b. Implement Delta Debugging, in your preferred programming language to minimize the input string that reveals the bug. Test your Delta Debugging code for the following input values provided.
        i. "abcdefG1"
        ii. "CCDDEExy"
        iii. "1234567b"
        iv. "8665"
   Briefly explain your delta-debugging algorithm and its implementation and provide the source code in/with your assignment. **(4 + 12 = 16 pts)**

6- Extra Credit Assignment: Create a GitHub repository to host all the elements of this assignment. This includes source codes, test data, and any screenshots or logs you have generated. Submit the GitHub link along with your main submission through Brightspace. **(5 pts)**

Marking Scheme: