CSC4001 Software Engineering

Assignment 2

Due – 23:59pm, 29th March 2023 (Wednesday)

Note:Late submission will have grade of 0!!!

Note: This assignment was designed, proofread, and released by teaching assistant Junjielong Xu (SDS, 222043010). If you have any question, please feel free to contact the TA.

There are a Python program and a Test Set below.
Please read carefully and answer all the following questions. (100 points)

Code:

```
1. def quicksort(array):
2.     less = []
3.     greater = []
4.     if len(array) < 2:
5.         return array
6.     else:
7.         pivot = array[0]
8.         for i in array[1:]:
9.             if i <= pivot:
10.                 less.append(i)
11.             else:
12.                 greater.append(i)
13.         return quicksort(less) + [pivot] + quicksort(greater)
```

Test Set: Test Case and Expected Result

1.	array = [1]	[1]
2.	array = [1, 2, 3, 4]	[1, 2, 3, 4]
3.	array = [4, 3, 2, 1]	[1, 2, 3, 4]
4.	array = [1, 1, 1, 1,]	[1, 1, 1, 1]
5.	array = [4, 5, 3, 1, 2]	[1, 2, 3, 4, 5]

To better represent code statements and test cases in the following questions, we use Line#N for the statement in N-th line, Case#N for N-th case, and Result#N for N-th Result. For example, Line#1 represents "def quicksort(array):" , Case#1 represents "array = [1]", and Result#1 represents "[1]"

P.S. The statement "detect the bugs by testing" does not mean that the exact line of code containing the bug needs to be identified, but rather it is sufficient to determine that "the code contains bugs."

Question 1 Mutation Testing (25 points)

Read the following mutants and answer the following question.

Specifically, please use Mutant#N to represent N-th mutant.

(e.g., "Δ 4. if len(array) >= 2:" means the mutant only changes the statement in Line#4 from "if len(array) < 2:" to "if len(array) >= 2:" compared with the origin program while the other lines remain the same.)

Mutants:

1. Δ 4. if len(array) >= 2:
2. Δ 7. pivot = array[-1]
3. Δ 8. for i in array:
4. Δ 9. if i > pivot:
5. Δ 13. return [pivot]

(a) Which mutants can be strongly killed by Case#1? Please provide the answer directly. (5 points)

(b) Which mutants can be strongly killed by Case#2? Please provide the answer directly. (5 points)

(c) Which mutants can be strongly killed by Case#3? Please provide the answer directly. (5 points)

(d) Please calculate the mutation score of Case#4 under strongly kill condition and write down the surviving mutants. (5 points)

(e) Please calculate the mutation score of Case#5 under strongly kill condition and write down the surviving mutants. (5 points)

Question 2 Differential Testing (20 points)

Read the following Python Code and answer the following question.

Note that both quicksort and bubblesort are sort algorithms. For the same sequence input, their output should normally be the same.

```
1. def bubblesort(array):
2.     n = len(array)
3.     for i in range(n):
4.         for j in range(0, n-i-1):
5.             if array[j] > array[j+1]:
6.                 array[j], array[j+1] = array[j+1], array[j]
7.     return array
```

(a) If we use Mutant#1 in **Question 1** to modify the program in the problem stem and obtain a quicksort with a bug, which test cases in the problem stem can successfully detect the code bugs in the modified quicksort? Please directly write the answer. (5 points)

(b) If we use Mutant#1 to Mutant#5 in **Question 1** in turn to modify the program in the problem stem and obtain a total of 5 quicksort programs with code bugs (which means we will have 5 quicksort programs with code bugs to be tested). Which test case(s) in the problem stem can successfully detect the most modified programs with bugs, and which Mutants' bug can be detected by this test case? Please provide the answers with some brief explanation. (15 points)

(If there are multiple qualified test cases, you need to write down the number of each of the buggy programs that can detect bugs; the corresponding buggy programs number is defined as the same number as Mutant # .)

Question 3 Metamorphic Testing (20 points)

Assuming the usage of the **inverse** metamorphic testing rule **g**, a new set of test cases can be automatically generated based on the given test cases from the problem stem. For example, **g(x)** represents the transformed test case of any given test case **x**

Herein, an example of the inverse metamorphic testing rule **g** for this problem is provided below.
Test case → Metamorphosed test case:
x: array = [num1, num2, num3,…,numN] → g(x): array = [numN, numN-1,…,num1]

Please answer the following questions by applying the metamorphic testing rule g in conjunction with the program specified in the problem stem.

(a) If the Result of the original test case is marked as **y=f (x)**, the converted test case is marked as **x '=g (x)**, and the Result of the converted test case is marked as **z=f (x')=f (g (x))**, please write the function of **y** on **z** directly (i.e., written in the form of **z=F (y)**), and provide all expected Results of all converted test cases **z**. (10 points)

(b) If we **only** use a certain test case and its transformed test case generated by the metamorphic testing rule **g** from Case#1~Case#5 **without Results** for testing, which test case groups can detect the bugs in the program generated by Mutant#4 in **Question 1**, which changes the quicksort in the problem stem to produce the buggy code? Please provide the answers (test case ID) with the explanation (i.e., why them or why not them). (10 points)

Question 4 Fuzzing (20 points)

Please provide a brief comparison of the similarities and differences between Mutation-Based Fuzzing and Generation-Based Fuzzing **(Less than 200 words)**

P.S. To accurately differentiate between the two Fuzzing methods, test case and test Result can be used as examples.

（This is an open-ended question, and a logical and well-reasoned answer is enough.）

Question 5 Reduction(15 points)

Please provide a brief overview of what problem **test case reduction** aim to solve and how **delta debugging** is implemented. **(Less than 200 words)**

（This is an open-ended question, and a logical and well-reasoned answer is enough.）