CSC4001 Software Engineering

Assignment 2

Due – 23:59pm, 15th March 2024 (Friday)

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 send emails to the TA.

There are a Python3 program and corresponding test set below.
Please read carefully and answer all the following questions. (100 points)

Program:

```
1. def caesar_cipher(s, b): # string and bias
2.     output = ""
3.     for char in s:
4.         ascii_val = ord(char)
5.         if 65 <= ascii_val <= 90:
6.             output += chr((ascii_val - 65 + b) % 26 + 65)
7.         elif 97 <= ascii_val <= 122:
8.             output += chr((ascii_val - 97 + b) % 26 + 97)
9.         else:
10.             output += char
11.     return output
```

(`ord` function maps character to ascii value, `chr` function maps ascii value to character.)

Test set: test input and expected output

1. ("abc", 1) "bcd"
2. ("A", 2) "C"
3. ("XYZ", -2) "VWX"
4. ("Khoor, Zruog!", -3) "Hello, World!"
5. ("Siyuexi", 3) "Vlbxhal"

To better represent code statements and test cases in the following questions, we use Line#N for the statement in N-th line, I#N for N-th input of the test case, and O#N for N-th expected output of the test case, Case#N for N-th input-output pair. For example, in the program above, Line#1 represents `def caesar_cipher(s, b):` , I#1 represents ("abc", 1), and O#1 represents "bcd"

P.S. The statement "test programs" or "detect the defects of the program 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 defects."

Question 1 Mutant Testing (30 points)

Read the following mutants and answer the following question.

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

(e.g., "Δ 5. if ascii_val <= 90:" means the mutant only changes the statement in Line#5 from "if 65 <= ascii_val <= 90:" to "if ascii_val <= 90:" compared with the origin program while the other lines remain the same.)

Mutants:

1. Δ 5. if ascii_val <= 90:
2. Δ 6. output = chr((ascii_val - 65 + b) % 26 + 65)
3. Δ 7. elif 97 < ascii_val < 122:
4. Δ 8. output += chr((ascii_val - 97 - b) % 26 + 97)
5. Δ 10. continue

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

(b) Which test cases can strongly kill Mutant#2? Please provide the answer directly. (5 points)

(c) Please calculate the mutation score when only consider Case#3 as a complete test suite under strongly kill condition. Please also write down the surviving mutants. (5 points)

(d) Please calculate the mutation score when considering Case#4 and Case#5 as a complete test suite under strongly kill condition. Please also write down the surviving mutants. (5 points)

(e) Please list all the mutant that <= 2 test cases that cannot strongly kill it. (5 points)

(f) Please list all the mutant that <= 1 test cases that can strongly kill it. (5 points)

Question 2 Differential Testing (20 points)

Read the following Python3 Code and answer the following question.

Here is another alternative implementation of the same caesar cipher algorithm using the List without `ord` or `chr`. For the same input, their output should normally be the same.

```
1. def caesar_cipher_alternative(s, b):
2.     lower_case_letters = 'abcdefghijklmnopqrstuvwxyz'
3.     upper_case_letters = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
4.     output = ""
5.
6.     for char in s:
7.         if char in lower_case_letters:
8.             idx = (lower_case_letters.index(char) + b) % 26
9.             output += lower_case_letters[idx]
10.        elif char in upper_case_letters:
11.            idx = (upper_case_letters.index(char) + b) % 26
12.            output += upper_case_letters[idx]
13.        else:
14.            output += char
15.    return output
```

(a) If the source code of `caesar_cipher_alternative` is accessible and known as bug-free, the source code of `caesar_cipher` is also accessible, and the expected output of all the test cases (O#1~O#5) is unknown. Is it **possible** to determine whether `caesar_cipher` have some bugs? (5 points)

(b) If the source code of `caesar_cipher_alternative` is accessible and known as bug-free, the source code of `caesar_cipher` is not accessible, and the expected output of all the test cases (O#1~O#5) is unknown. Is it **possible** to determine whether `caesar_cipher` have some bugs? (5 points)

(c) If the source code of `caesar_cipher_alternative` is not accessible but known as bug-free, the implementation of `caesar_cipher` is not accessible, and the expected output of all the test cases (O#1~O#5) is unknown. Is it **possible** to determine whether `caesar_cipher` have some bugs? (5 points)

(d) If the implementations of `caesar_cipher_alternative` and `caesar_cipher` are not accessible, and the expected output of all the test cases (O#1~O#5) is unknown. Is it **possible** to determine whether `caesar_cipher` have some bugs? (5 points)

p.s. “the source code is not accessible” means that you can still call this function, but you do not know the details of the source code. i.e., the function is a black box.

Question 3 Metamorphic Testing (25 points)

We use the notation **g(x)** to represent the transformed test case input of any given test case input **x**. In this question, we provide a metamorphic relation as below:

Test case input → Metamorphosed test case input:
x: (s, b) → g(x): (s, b+26)

Please answer the following questions by applying the metamorphic testing rule **g** in the program shown in the first page.

(a) If the expected output O#1~O#5 of the original test set is marked as **y=f (x)** (f means the abstract mapping function of the program), the transformed test case input is marked as **x '=g (x)**, and the output of the transformed test case input is marked as **z=f (x')=f (g (x))**, please write the function of **y** on **z**. (i.e., written in the form of **z=F (y)**. You need to specify the mapping "F") (10 points)

(b) If we **only** use a certain test case and its transformed test case generated by the metamorphic testing rule **g** from I#1~I#5 **without expected output** for testing, which test cases can detect the defects in the modified program by adopting Mutant#4 in **Question 1**? Please provide the answers (test case ID) with the explanation (i.e., why them or why not them). (15 points)

Question 4 Intramorphic Testing (25 points)

Assuming that we obtain a new caesar cipher program by inserting a line of code:

```
b = -(26 - b) % 26
```

between Line#1 and Line#2 in the program specified in the problem stem, please answer the following questions based on the origin program (called **P**) and the modified program (called **P'**).

(a) Please provide the expected output of the test case input I#1~I#5 for the five test cases in the problem stem in **P'**. (10 points)

(b) If Mutant #4 from **Question 1** is used to alter the original program **P** and the modified program **P',** which test cases can detect the defects in the modified program **without expected output**? Please provide the answers (test case ID) with the explanation (i.e., why them or why not them). (15 points)