Testing und Debugging - Übungsblatt 6:

Gruppe 11:
Manuel Neubauer - 79358
Hueseyin Selcuk - 79375
Abdullah Yildiz - 79669
Daniel Zahl - 79356

1. **Verstehen des internen Aufbaus und der Benutzung des Testfallgenerators**

    1. Aufgabe 1:

    Ein Pseudozufallsgenerator kann über eine Software programmiert werden. Ein echter Zufallsgenerator ist schwer zu realisieren, und es wird extra Hardware benötigt (z.B. DRNG von Intel).

    Der Vorteil von PRNG's (pseudo random number generator) ist die hohe Geschwindigkeit und die Verfügbarkeit in praktisch allen Programmiersprachen. Der Grund für die Schnelligkeit liegt daran, dass PRNG's deterministisch sind. Der PZG kann bei Angabe des Definitionsbereichs auch nur gewünschte Datentypen, wie Integer, Strings etc. erzeugen.
    Der PZG erzeugt immer wieder die selbe Sequenz von zufälligen Werten, wenn der Seed gleich bleibt.

    2. Aufgabe 2:

    Ja, es ist möglich. Wichtig dabei ist, das die Aufrufparameter nicht verändert werden:

```
create_random_api_calls(api= [“empty”, “full”, “enqueue”, “dequeue”],
                                        queues=["MyQueue6"],
                                        queue_size=1983,
                                        max_queue_size=2000,
                                        stream_length=5267,
                                        max_stream_length=16000,
                                        seed=13681756363766)
```

    Die Werte lassen sich aus der Datei „API_test_stream“ ermitteln.

    Die queue_size kann man von anhand der Eingabe ablesen: “MyQueue6 1983”
    (ausgewählte Queue und die Länge, Z:93 uebung06.py)

    Die Werte “max_queue_size”, “max_stream_length" waren im API_test_stream auf “None” gesetzt. Dadurch könnte man die Werte auch leer lassen, sonst müsste man ebenfalls die exakten Werte übernehmen.

2. **Erst Denken, dann Ausprobieren**

1. Aufgabe 1:

Der Fehler kommt erst zustande, wenn MyQueue3 mehr als 256 Elemente erhält.
In der MyQueue.py steht in Z:101 folgendes:

min(self.max, 2 ** 8) = min(self.max, 256)

Da die Queuesize standardmäßig als Zufallswert zwischen 1 und max_queue_size(2000) liegt. Der Fehler tritt erst bei einer Size von 256 auf. Es ist somit recht unwahrscheinlich, dass die Queue dieses Testfalles mit einer ausreichend großen queuesize initialisiert wird.
Damit der Fehler in MyQueue3 ausgelöst wird muss die Queue mit 2^8 (256) Elementen aufgefüllt werden.Die Wahrscheinlichkeit, dass die API-Calls genau in der Reihenfolge erfolgen, dass die Queue mit 256 Elementen voll wird ist recht unwahrscheinlich, da ja auch zwischendurch wieder dequeue aufgerufen werden kann und so die Füllung der Queue kleiner wird.

2. Aufgabe 2:

MyQueue3 257
256 x Enqueue [Int]
1 x Full

3. Aufgabe 3:

Aufruf:

```
create_random_api_calls(api=[“full”, “enqueue”],
                        queues=[“MyQueue3”],
                        queue_size=1000,
                        stream_length=None,
                        max_stream_length = 16000
                        seed=13681756363766)
```

Der API Aufruf hat kein “dequeue”, da wir keine Elemente entfernen wollen. Ebenfalls ist “empty” nicht nötig, da wir nicht auf leere Listen prüfen. Durch die Übergabe eines bestimmten Seeds ist dieselbe API-Aufrufsequenz beliebig oft nachstellbar. Da die queue_size über 256 ist, wird jedes mal der Fehler reproduziert.