Quiz 5

學號: 111550129　姓名: 林彥亨

Problem 1

In this homework, you need to download the Random Number Generator Test Suite from the US National Institute of Standards and Technology (NIST) to ensure that the random numbers you choose are truly unpredictable and secure.

a) Write a Python/C++ program to generate **1M bytes** of cryptographically secure random numbers.

b) Run the NIST SP 800-22 statistical test on your **1M bytes** of binary cryptographically secure random numbers and analyze the test results to identify any deviations from the expected statistical properties of random numbers.

c) Extra credit: Find out a non-cryptographically secure random number generator, such as random(), to demonstrate its lack of safety. Then, propose modifications to enhance its security to generate cryptographically secure random numbers that meet the highest standards of security and reliability.

(a)

- Here is our python code.

```python
import secrets
# the function to generate the secure random number
def generate_randomNum(num_bits: int):
    random_bits = secrets.token_bytes(num_bits)
    # return ''.join(f'{byte:08b}'for byte in random_bytes)
    return random_bits
# generate 1M bytes random number
random_bits = generate_randomNum(8388608) # 8388608 = 1024 * 1024 * 8 (bits)
# output a file 'random.bin'
with open('random.bin', 'wb') as file:
    file.write(random_bits)
```

  - First, we import the library of “secretes”
  - Then we construct a function to generate the random number.
  - Finally, let the output to be a “random.bin” file.

(b)

We prepared for the testing file, which is required binary sequences as input in the test suite. In order to running the NIST SP 800-22, we using the Ubuntu to run the test suite. And we run in tenminal.

```
heng@heng-VirtualBox:~/下載/sts-2_1_2/sts-2.1.2/sts-2.1.2$ ./assess 8388608
          G E N E R A T O R    S E L E C T I O N
          ---------------------------------

   [0] Input File                 [1] Linear Congruential
   [2] Quadratic Congruential I   [3] Quadratic Congruential II
   [4] Cubic Congruential         [5] XOR
   [6] Modular Exponentiation     [7] Blum-Blum-Shub
   [8] Micali-Schnorr             [9] G Using SHA-1

  Enter Choice: 0


              User Prescribed Input File: random.bin
```

```
            S T A T I S T I C A L   T E S T S
            -------------------------------

  [01] Frequency                       [02] Block Frequency
  [03] Cumulative Sums                 [04] Runs
  [05] Longest Run of Ones             [06] Rank
  [07] Discrete Fourier Transform      [08] Nonperiodic Template Matchings
  [09] Overlapping Template Matchings  [10] Universal Statistical
  [11] Approximate Entropy             [12] Random Excursions
  [13] Random Excursions Variant       [14] Serial
  [15] Linear Complexity

       INSTRUCTIONS
          Enter 0 if you DO NOT want to apply all of the
          statistical tests to each sequence and 1 if you DO.

 Enter Choice: 1
```

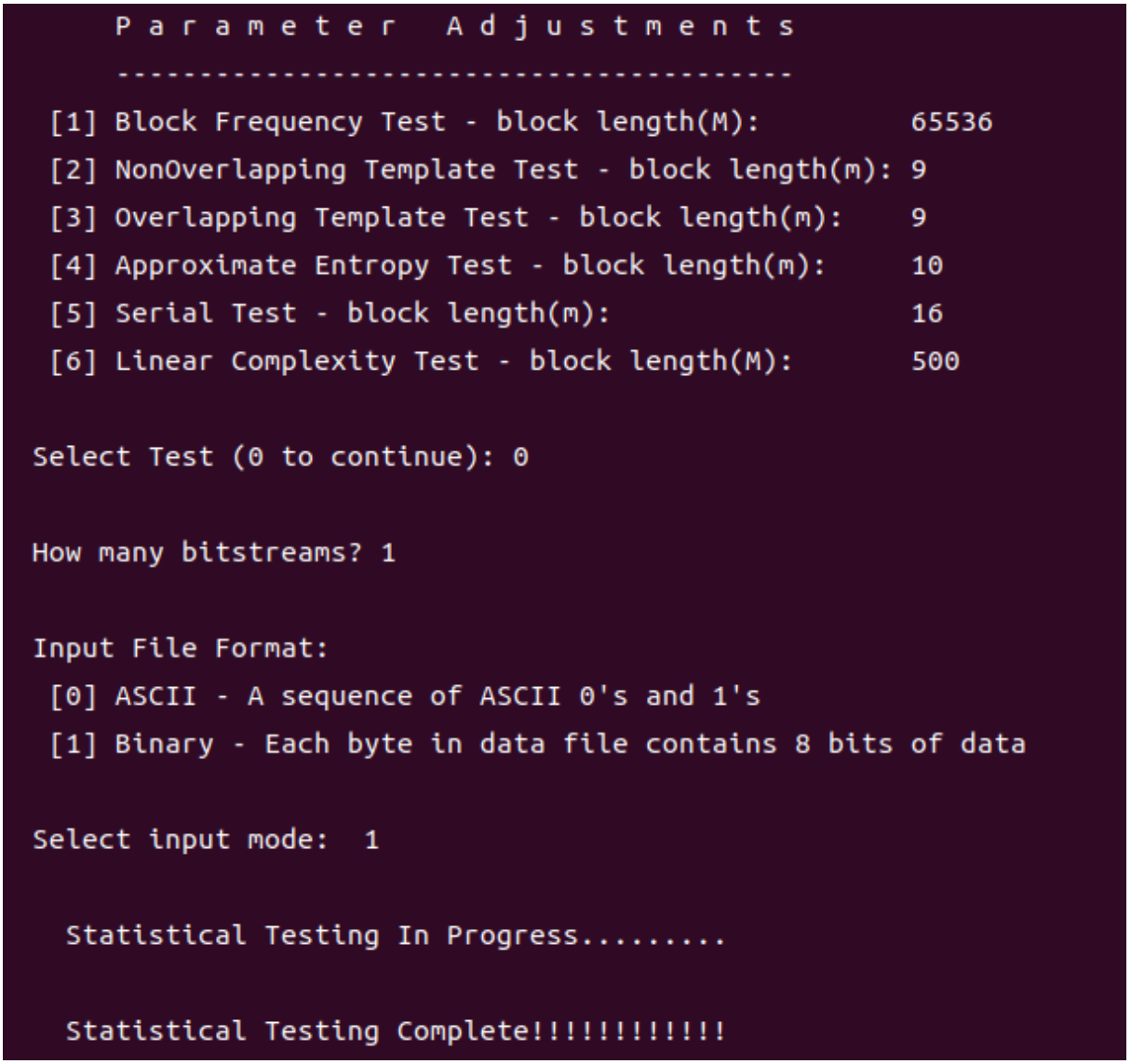
```
     P a r a m e t e r   A d j u s t m e n t s
     ----------------------------------------
 [1] Block Frequency Test - block length(M):         128
 [2] NonOverlapping Template Test - block length(m): 9
 [3] Overlapping Template Test - block length(m):    9
 [4] Approximate Entropy Test - block length(m):     10
 [5] Serial Test - block length(m):                  16
 [6] Linear Complexity Test - block length(M):       500

Select Test (0 to continue): 1

Enter Block Frequency Test block length: 65536

     P a r a m e t e r   A d j u s t m e n t s
     ----------------------------------------
 [1] Block Frequency Test - block length(M):         65536
 [2] NonOverlapping Template Test - block length(m): 9
 [3] Overlapping Template Test - block length(m):    9
 [4] Approximate Entropy Test - block length(m):     10
 [5] Serial Test - block length(m):                  16
 [6] Linear Complexity Test - block length(M):       500

Select Test (0 to continue): 0
```



Above is the step i run in the terminal.

The result and analyze please refer to the “finalAnalysisReport.txt”.