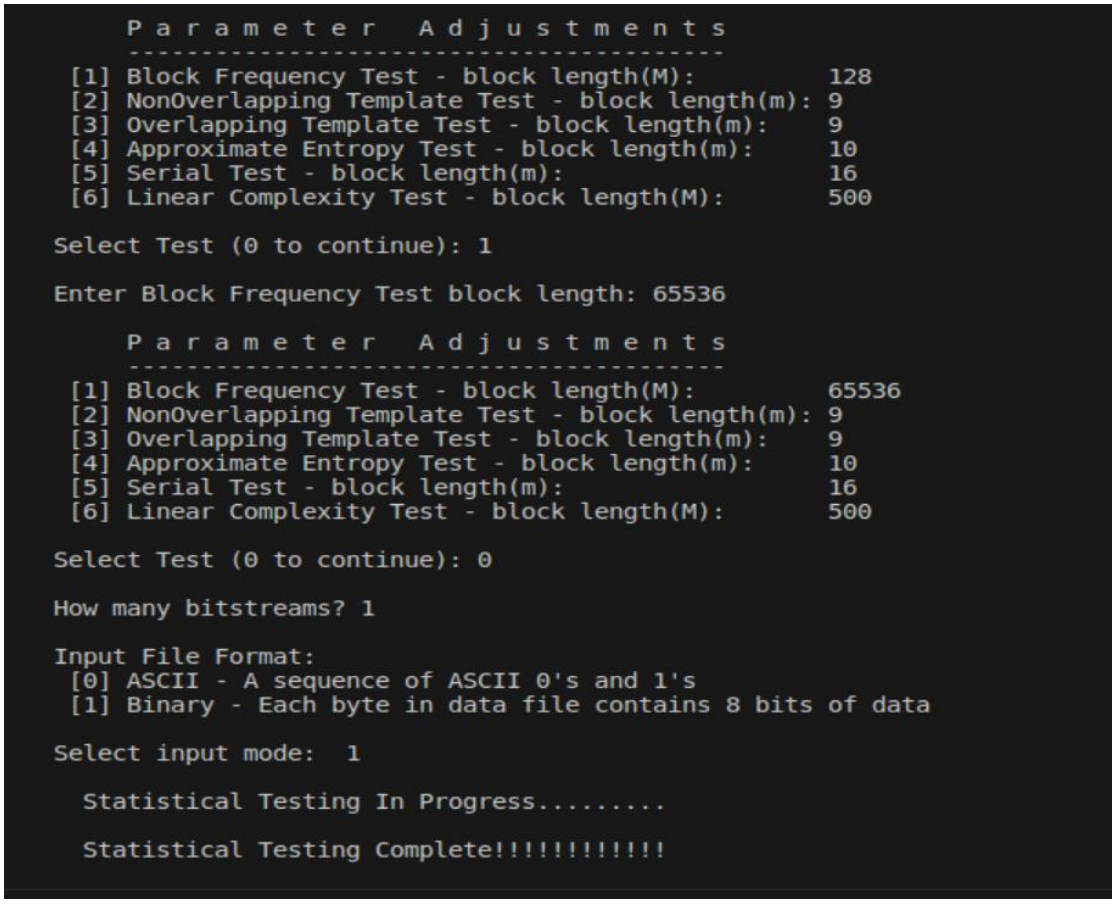
1

```python
import os
random_data = os.urandom(1024*1024)
with open('random.bin', 'wb') as file:
    file.write(random_data)

```

2

```
yang@yang-VirtualBox:~/桌面/mimago/ubuntuuu/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: ran.bin


              User Prescribed Input File:
              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
```



(a) Frequency (Monobit) Test: This test checks if there's a roughly equal number of ones and zeros in the sequence, which is a trait of randomness.

(b) Block Frequency Test: This one looks at smaller blocks of the sequence to see if the count of ones in each block matches what you'd expect from random chance, typically half the block size.

(c) Runs Test: This test observes the sequence for runs, which are uninterrupted sequences of identical bits (ones or zeros), and measures if their number and length match what a random sequence would typically have.

(d) Longest Run of Ones in a Block Test: It identifies the longest string of ones in a block and checks if this length makes sense for a random sequence.

(e) Binary Matrix Rank Test: This test searches for patterns that could suggest the sequence isn't really random by looking at linear relationships between parts of the sequence.

(f) Discrete Fourier Transform (Spectral) Test: This checks for repeating patterns at regular intervals in the sequence that might hint it's not truly random.

(g) Non-overlapping Template Matching Test: It looks for too many occurrences of a certain complex pattern, which could indicate a lack of randomness.

(h) Overlapping Template Matching Test: Similar to the non-overlapping version, but it searches for a specific pattern in a sliding window that moves one bit after each check.

(i) Maurer's Universal Statistical Test: It assesses if the sequence could be significantly compressed, which should not be possible with truly random data.

(j) Linear Complexity Test: This test measures the complexity of the sequence to see if it's complicated enough to be considered random.

(k) Serial Test: It checks if the counts of all possible overlapping patterns of a certain length are close to what would be expected from a random sequence.

(l) Approximate Entropy Test: This one compares how often blocks of two sizes (m and m+1) occur, looking for the variety expected in a random sequence.

(m) Cumulative Sums (Cusums) Test: It evaluates whether the cumulative total of the sequence's values swings too high or too low compared to what randomness would dictate.

(n) Random Excursions Test: This test looks at how often the sequence visits particular values and compares this to the expected frequency for a random sequence.

(o) Random Excursions Variant Test: It's similar to the Random Excursions Test but focuses on the number of times the sequence reaches different states, checking for randomness.