密碼工程 Quiz5

a

code:

```python
import secrets

def generate_random_bytes(num_bytes):
    return bytes([secrets.randbits(8) for _ in range(num_bytes)])

def main():
    num_bytes = 1024 * 1024  # 1 million bytes
    random_bytes = generate_random_bytes(num_bytes)

    # Write random bytes to a file
    with open("random.bin", "wb") as f:
        f.write(random_bytes)

    print("Random bytes generated successfully.")

if __name__ == "__main__":
    main()
```

→ First of all, I import "secrets". Next, I define the "generate_random_bytes" function, which utilizes the "randbits" function in "secrets" to create bits randomly. Subsequently, convert this list of bits into a bytes object.
In the main function, I set the size of bytes to 1024*1024, which is required in the problem a. Then, use the "generate_random_bytes" function I defined to generate the random bytes and store it in the random.bin. If the funcion implements successfully, then print the message, "Random bytes generated successfully".

result:

```
Random bytes generated successfully.
```

b

The result of running the NISTSP800-22 statistical test on my 1M bytes of binary cryptographically secure random numbers:

```
ysisReport.txt
---------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
---------------------------------------------------------------------------------
   generator is <random.bin>
---------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
---------------------------------------------------------------------------------
  0   0   0   0   0   0   0   1   0   0     ----       1/1       Frequency
  0   0   0   0   0   0   1   0   0   0     ----       1/1       BlockFrequency
  0   0   0   0   0   0   0   0   0   1     ----       1/1       CumulativeSums
  0   0   0   0   0   0   1   0   0   0     ----       1/1       CumulativeSums
  0   1   0   0   0   0   0   0   0   0     ----       1/1       Runs
  0   0   0   0   1   0   0   0   0   0     ----       1/1       LongestRun
  0   0   0   0   1   0   0   0   0   0     ----       1/1       Rank
  0   0   0   0   0   0   0   0   0   1     ----       1/1       FFT
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   1   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   1   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   0   1     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   1   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   1   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   1   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   1   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   1   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   1   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   1   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  1   0   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   1   0   0     ----       1/1       NonOverlappingTemplate
  0   0   1   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   1   0     ----       1/1       NonOverlappingTemplate
  0   1   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   0   1     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   1   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   0   1     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  1   0   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   1   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   1   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   1   0     ----       1/1       NonOverlappingTemplate
  1   0   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   0   1     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   1   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   1   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   0   1     ----       1/1       NonOverlappingTemplate
  0   0   1   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   1   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   1   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   0   0   0   0   1     ----       1/1       NonOverlappingTemplate
  1   0   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   1   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  1   0   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   0   0   1   0   0   0   0     ----       1/1       NonOverlappingTemplate
  1   0   0   0   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
  0   0   0   1   0   0   0   0   0   0     ----       1/1       NonOverlappingTemplate
```

```
 0  0  0  0  0  0  0  1  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  0  1    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  1  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  0  1    ----      1/1      NonOverlappingTemplate
 0  0  0  1  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  1  0  0  0    ----      1/1      NonOverlappingTemplate
 0  1  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  1  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  1  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  0  1    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  0  1    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  0  1    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  1  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  1  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  1  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  1  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  1  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  0  1    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  1  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  1  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  1  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  1  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  1  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  1  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  1  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  1  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  1  0  0    ----      1/1      NonOverlappingTemplate
 0  0  1  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 1  0  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  1  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  1  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  1  0  0  0  0  0  0  0  0    ----      1/1      NonOverlappingTemplate
 0  0  0  0  0  0  0  0  1  0    ----      1/1      NonOverlappingTemplate
```

```
 0   0   1   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   1   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   1   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   0   1   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   0   0   1     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   0   1   0     ----      1/1       NonOverlappingTemplate
 0   0   1   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 1   0   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   1   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   1   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   1   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   0   0   1     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   1   0   0     ----      1/1       NonOverlappingTemplate
 1   0   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   1   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   1   0   0   0     ----      1/1       NonOverlappingTemplate
 0   1   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   1   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   1   0   0     ----      1/1       NonOverlappingTemplate
 1   0   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   1   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   1   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   0   0   0   0   0   0   0   0   1     ----      1/1       NonOverlappingTemplate
 0   1   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   1   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 1   0   0   0   0   0   0   0   0   0     ----      1/1       NonOverlappingTemplate
 0   1   0   0   0   0   0   0   0   0     ----      1/1       OverlappingTemplate
 0   0   0   1   0   0   0   0   0   0     ----      1/1       Universal
 0   0   0   0   0   1   0   0   0   0     ----      1/1       ApproximateEntropy
 0   1   0   0   0   0   0   0   0   0     ----      1/1       RandomExcursions
 0   0   0   0   1   0   0   0   0   0     ----      1/1       RandomExcursions
 0   0   0   0   0   1   0   0   0   0     ----      1/1       RandomExcursions
 0   1   0   0   0   0   0   0   0   0     ----      1/1       RandomExcursions
 0   0   0   0   0   0   0   1   0   0     ----      1/1       RandomExcursions
 0   0   0   0   0   1   0   0   0   0     ----      1/1       RandomExcursions
 0   0   0   0   0   1   0   0   0   0     ----      1/1       RandomExcursions
 1   0   0   0   0   0   0   0   0   0     ----      1/1       RandomExcursions
 0   0   1   0   0   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   1   0   0   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   1   0   0   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   1   0   0   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   1   0   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   0   0   0   0   1     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   0   1   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   0   0   1   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   1   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   0   0   0   1   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   1   0   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   1   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   0   0   0   1   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   0   0   1   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   1   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   0   0   0   0   1   0   0   0   0     ----      1/1       RandomExcursionsVariant
 0   1   0   0   0   0   0   0   0   0     ----      1/1       Serial
 1   0   0   0   0   0   0   0   0   0     ----      1/1       Serial
 0   0   1   0   0   0   0   0   0   0     ----      1/1       LinearComplexity


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 0 for a
sample size = 1 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 0 for a sample size = 1 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
```

There are some tests in theNISTSP800-22 statistical test:

1. Frequency:
   First of all, convert the 0 in sequence to -1 and 1 in sequence to -1, and then add them together to produce S_n.
   Secondly, compute the test statistic $s_{obs}= \frac{|S\ n\ |}{\sqrt{n}}$.
   Finally, we utilize the s_{obs} to compute the P-value.
   P-value = **erfc**($\frac{S\ obs}{\sqrt{2}}$)
   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

2. Block Frequency:
   First of all, Partition the input sequence into N = $\lfloor \frac{n}{M} \rfloor$ non-overlapping blocks. Discard any unused bits.
   Secondly, Determine the proportion π_i of ones in each M-bit block using the equation

$$\pi_i = \frac{\sum_{j=1}^{M} \varepsilon_{(i-1)M+j}}{M}$$

   Thirdly, compute the χ^2 statistic:

$$\chi^2(obs) = 4M \sum_{i=1}^{N} (\pi_i - 1/2)^2$$

   Finally, compute the P-value.

$$P\text{-}value = \mathbf{igamc}(N/2, \chi^2(obs)/2)$$

   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

3. CumulativeSums:
   First of all, convert the 0 in sequence to -1 and 1 in sequence to -1.
   Secondly, Compute partial sums S_i of successively larger subsequences, each starting with X_1 (if mode=0) or X_n (if mode=1).
   Thirdly, Compute the test statistic $z = \max_{1 \le k \le n} |S_k|$
   Finally, compute P-value.

$$P\text{-}value = 1 - \sum_{k=\left(\frac{-n}{z}+1\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right) \right] +$$

$$\sum_{k=\left(\frac{-n}{z}-3\right)/4}^{\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) \right]$$

   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

4. Runs:
   First of all, Compute the pre-test proportion π of ones in the input sequence:

$$\pi = \frac{\sum_j \varepsilon_j}{n}$$

   Secondly, determine if the prerequisite frequency test is passed. If it passed, then continue the test. Otherwise, just stop it.
   Thirdly, Compute the test statistic $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$
   Finally, compute P-value.

$$P\text{-}value = \mathbf{erfc}\left(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right)$$

   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

5. LongestRun:
   First of all, Divide the sequence into M-bit blocks.
   Secondly, tabulate the frequencies ν_i of the longest runs of ones in each block into categories, where each cell contains the number of runs of ones of a given length.

ν_i	M = 8	M = 128	$M = 10^4$
ν_0	≤ 1	≤ 4	≤ 10
ν_1	2	5	11
ν_2	3	6	12
ν_3	≥ 4	7	13
ν_4		8	14
ν_5		≥ 9	15
ν_6			≥ 16

   Thirdly, compute $\chi^2(obs) = \sum_{i=0}^{K} \frac{(\nu_i - N\pi_i)^2}{N\pi_i}$. The values of K and N are determined by the value of M in accordance with the following table:

M	K	N
8	3	16
128	5	49
10^4	6	75

   Finally, compute P-value.

$$P\text{-}value = \mathbf{igamc}\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right)$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

6. Rank:
   First of all, sequentially divide the sequence into M•Q-bit disjoint blocks; there will exist N = $\lfloor \frac{n}{MQ} \rfloor$ such blocks. Discarded bits will be reported as not being used in the computation within each block. Collect the M•Q bit segments into M by Q matrices. Each row of the matrix is filled with successive Q-bit blocks of the original sequence ε.
   Secondly, Determine the binary rank (R_I) of each matrix.
   Thirdly, Let F_M = the number of matrices with R_I = M (full rank),
   F_{M-1} = the number of matrices with R_I = M-1 (full rank -1),
   N – F_M- F_{M-1} = the number of matrices remaining.
   Fourthly, compute:

$$\chi^2(obs) = \frac{(F_M - 0.2888N)^2}{0.2888N} + \frac{(F_{M-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_M - F_{M-1} - 0.1336N)^2}{0.1336N}$$

   Finally, compute P-value = $e^{-\chi^2(obs)/2}$

   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

7. FFT
   First of all, convert the 0 in sequence to -1 and 1 in sequence to -1.
   Secondly, Apply a Discrete Fourier transform (DFT) on X to produce: S = DFT(x). A sequence of complex variables is produced which represents periodic components of the sequence of bits at different frequencies.
   Thirdly, Calculate M = modulus(S´) ≡ |S'|, where S' is the substring of the first n/2 elements in S.
   Fourthly, compute $T = \sqrt{\left(\log\frac{1}{0.05}\right)n}$ = the 95% peak height threshold value.
   Fifthly, compute $N_0 = \frac{0.95n}{2}$ and compute N_1 = the actual observed number of peaks in M that are less than T.
   Sixthly, compute $d = \frac{(N_1 - N_0)}{\sqrt{n(.95)(.05)/4}}$

Finally, compute $P\text{-}value = erfc\left(\frac{|d|}{\sqrt{2}}\right)$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

8. Non-Over lapping Template:
   First of all, Partition the sequence into N independent blocks of length M.
   Secondly, let W_j(j=1,...,N) be the number of times that B (the template) occurs within the block j.
   Thirdly, Under an assumption of randomness, compute the theoretical mean μ and variance σ^2.

   $\mu = (M\text{-}m+1)/2^m$ $\qquad$ $\sigma^2 = M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right)$.

   Fourthly, compute $\chi^2(obs) = \sum_{j=1}^{N} \frac{(W_j - \mu)^2}{\sigma^2}$

   Finally, compute $P\text{-}value = \mathbf{igamc}\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right)$

   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

9. Over lapping Template:
   First of all, Partition the sequence into N independent blocks of length M.
   Secondly, Calculate the number of occurrences of B in each of the N blocks.
   Thirdly, compute values for λ and η that will be used to compute the theoretical probabilities π_i corresponding to the classes of v_0:
   $\lambda = (M\text{-}m+1)/2^m$ $\qquad$ $\eta = \lambda/2$.

   Fourthly, compute $\chi^2(obs) = \sum_{i=0}^{5} \frac{(v_i - N\pi_i)^2}{N\pi_i}$

   Finally, compute $P\text{-}value = \mathbf{igamc}\left(\frac{5}{2}, \frac{\chi^2(obs)}{2}\right)$

   If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

10. Universal:
    First of all, the n-bit sequence (ε) is partitioned into two segments : an initialization segment consisting of Q L-bit non-overlapping blocks, and a test segment consisting of K L-bit non-overlapping blocks. Bits remaining at the end

of the sequence that do not form a complete L-bit block are discarded.The first Q blocks are used to initialize the test. The remaining K blocks are the test blocks.
Secondly, Using the initialization segment, a table is created for each possible L-bit value. The block number of the last occurrence of each L-bit block is noted in the table.
Thirdly, examine each of the K blocks in the test segment and determine the number of blocks since the last occurrence of the same L-bit block. Replace the value in the table with the location of the current block. Add the calculated distance between re-occurrences of the same L-bit block to an accumulating $\log_2$ sum of all the differences detected in the K blocks.

Fourthly, compute the test statistic: $f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K} log_2(i - T_j)$

Finally, compute P-value.

$$P\text{-}value = \mathbf{erfc}\left(\left|\frac{f_n - expectedValue(L)}{\sqrt{2}\sigma}\right|\right)$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

11. Approximate Entropy:
First of all, augment the n-bit sequence to create n overlapping m-bit sequences by appending m-1 bits from the beginning of the sequence to the end of the sequence.
Secondly, a frequency count is made of the n overlapping blocks. Let the count of the possible m-bit ((m+1)-bit) values be represented as C_i^m, where i is the m-bit value.

Thirdly, compute $C_i^m = \frac{\#i}{n}$ for each value of i.

Fourthly, compute $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i log\ \pi_i$, where $\pi_i = C^3_j$, and $j=log_2 i$

Fifth, repeat step 1~4, replacing m by m+1.
Sixth, compute the test statistic

$\chi^2 = 2n[log\ 2 - ApEn(m)]$, where $ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$

Finally, compute P-value.

$$P\text{-}value = \mathbf{igamc}(2^{m-1}, \frac{\chi^2}{2})$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

12. Random Excursions
First of all, convert the 0 in sequence to -1 and 1 in sequence to -1.
Secondly, Compute the partial sums S_i of successively larger subsequences, each starting with X_1. Form the set S = {Si}.

$S_1=X_1$
$S_2 = X_1 + X_2$
$S_3 = X_1 + X_2 + X_3$
.
.
$S_k= X_1 + X_2 + X_3 + ... + X_k$
.
.
$S_n = X_1 + X_2 + X_3 + ... + X_k + ... + X_n$
Thirdly, Form a new sequence S' by attaching zeros before and after the set S. That is, S' = 0, s_1, s_2, ... , s_n, 0.
Fourthly, let J = the total number of zero crossings in S', where a zero crossing is a value of zero in S' that occurs after the starting zero. J is also the number of cycles in S', where a cycle of S' is a subsequence of S'consisting of an occurrence of zero, followed by no-zero values, and ending with another zero. The ending zero in one cycle may be the beginning zero in another cycle. The number of cycles in S'is the number of zero crossings. If J < 500, discontinue the test.
Fifth, for each cycle and for each non-zero state value x having values -4 ≤ x ≤ -1 and 1 ≤ x ≤ 4, compute the frequency of each x within each cycle.
Sixth, for each of the eight states of x, compute $v_k(x)$ = the total number of cycles in which state x occurs exactly k times among all cycles, for k = 0, 1, ... , 5.
Seventh, For each of the eight states of x, compute the test statistic

$$\chi^2(obs) = \sum_{k=0}^{5} \frac{(v_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$$

Finally, for each state of x, compute P-value.

$$P\text{-}value = \mathbf{igamc}(5/2,\ \chi^2(obs)/2)$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

13. Random Excursions Variant:
    First of all, convert the 0 in sequence to -1 and 1 in sequence to -1.
    Secondly, Compute the partial sums S_i of successively larger subsequences, each starting with X_1. Form the set S = {Si}.
    $S_1=X_1$
    $S_2 = X_1 + X_2$
    $S_3 = X_1 + X_2 + X_3$
    .
    .
    $S_k= X_1 + X_2 + X_3 + ... + X_k$
    .
    .
    $S_n = X_1 + X_2 + X_3 + ... + X_k + ... + X_n$
    Thirdly, Form a new sequence S' by attaching zeros before and after the set S.

That is, S' = 0, s_1, s_2, … , s_n, 0.
Fourth, for each of the eighteen non-zero states of x, compute $\xi(x)$ = the total number of times that state x occurred across J cycles.
Finally, for each $\xi(x)$, compute P-values.

$$P\text{-}value = \mathbf{erfc}\left(\frac{|\xi(x) - J|}{\sqrt{2J(4|x| - 2)}}\right)$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

14. Serial:
First of all, extend the sequence by appending the first m-1 bits to the end of the sequence for distinct values of n to get the augmented sequence ε'.
Secondly, determine the frequency of all possible overlapping m-bit blocks, all possible overlapping (m-1)-bit blocks and all possible overlapping (m-2)-bit blocks. Let $v_{i_1 \ldots i_m}$ denote the frequency of the m-bit pattern i_1 … i_m.

$$\text{Compute: } \psi_m^2 = \frac{2^m}{n}\sum_{i_1 \ldots i_m}\left(v_{i_1 \ldots i_m} - \frac{n}{2^m}\right)^2 = \frac{2^m}{n}\sum_{i_1 \ldots i_m} v_{i_1 \ldots i_m}^2 - n$$

Thirdly,

$$\psi_{m-1}^2 = \frac{2^{m-1}}{n}\sum_{i_1 \ldots i_{m-1}}\left(v_{i_1 \ldots i_{m-1}} - \frac{n}{2^{m-1}}\right)^2 = \frac{2^{m-1}}{n}\sum_{i_1 \ldots i_{m-1}} v_{i_1 \ldots i_{m-1}}^2 - n$$

$$\psi_{m-2}^2 = \frac{2^{m-2}}{n}\sum_{i_1 \ldots i_{m-2}}\left(v_{i_1 \ldots i_{m-2}} - \frac{n}{2^{m-2}}\right)^2 = \frac{2^{m-2}}{n}\sum_{i_1 \ldots i_{m-2}} v_{i_1 \ldots i_{m-2}}^2 - n$$

Fourthly,

$$\text{Compute: } \nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2, \text{ and}$$
$$\nabla^2\psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2.$$

Finally, compute P-value

$$P\text{-}value1 = \mathbf{igamc}\left(2^{m-2}, \nabla\psi_m^2\right) \text{ and}$$

$$P\text{-}value2 = \mathbf{igamc}\left(2^{m-3}, \nabla^2\psi_m^2\right).$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

15. Linear Complexity:
First of all, partition the n-bit sequence into N independent blocks of M bits, where n=MN.
Secondly, using the Berlekamp-Massey algorithm, determine the linear complexity L_i of each of the N blocks (i = 1, … ,N). L_i is the length of the shortest linear feedback shift register sequence that generates all bits in the block i.
Within any L-bit sequence, some combination of the bits, when added together modulo 2, produces the next bit in the sequence (bit L; + 1).
Thirdly, under an assumption of randomness, calculate the theoretical mean μ:

$$\mu = \frac{M}{2} + \frac{\left(9 + (-1)^{M+1}\right)}{36} - \frac{\left(M/3 + 2/9\right)}{2^M}.$$

Fourth, for each substring, calculate a value of T_i, where

$$T_i = (-1)^M \bullet (L_i - \mu) + 2/9$$

Fifth, record the T_i values in $v_0,\ldots,v_6$ as follows:

If:	$T_i \le -2.5$	Increment v_o by one
	$-2.5 < T_i \le -1.5$	Increment v_1 by one
	$-1.5 < T_i \le -0.5$	Increment v_2 by one
	$-0.5 < T_i \le 0.5$	Increment v_3 by one
	$0.5 < T_i \le 1.5$	Increment v_4 by one
	$1.5 < T_i \le 2.5$	Increment v_5 by one
	$T_i > 2.5$	Increment v_6 by one

Sixth, compute $\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$

Finally, compute P-value.

$$P\text{-}value = \mathbf{igamc}\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right)$$

If P-value < 0.01, then conclude that the sequence is non-random. Otherwise, conclude that the sequence is random.

C

I use "pip install pycryptodome" to install the library I need in bonus.py (http://bonus.py). At the beginning, I use randint in random library to generate the non-cryptographically secure random number and store in random_bonus.bin. Next, I made random_bonus.bin tested in NIST and store result in finalAnalysisReport_random.txt. As expected at beginning, some tests conducted at NIST have failed to pass. Subsequently, I utilized the encrypt function to perform AES encryption on the random number I generated earlier, storing it in random_bonus_AES.bin. Similarly, I also made random_bonus_AES.bin tested in NIST and store result in finalAnalysisReport_random_AES.txt. This time, it passed all the tests. Therefore, we can confirmed that this random number is now cryptographically secure.