Problem 1

1. Comparison of SHA-256 and SHA-512/256:

- Structure

	SHA-256	SHA-512/256
基本架構	Merkle–Damgård	Merkle–Damgård
輸入區塊大小	512 bits	1024 bits
內部狀態大小	256 bits	512 bits
初始常數	取自前 8 個質數的平方根小數（32 bits）	取自前 8 個質數的立方根小數（64 bits）
計算輪數	64	80
輸出長度	256 bits	256 bits（由 SHA-512 截斷而來）
平台適應性	適合 32-bit 系統	適合 64-bit 系統（效能更佳）

- Security properties

	SHA-256	SHA-512/256
碰撞抵抗性（Collision Resistance）	128-bit（需約 2^{128} 次操作）	相同（因為截斷後也是 256 bits 輸出）
預映像抵抗性（Preimage Resistance）	$\approx 2^{256}$	相同
第二預映像抵抗性（Second Preimage Resistance）	$\approx 2^{256}$	相同
延伸攻擊（Length Extension Attack）	可能遭受攻擊（需額外保護）	同樣有風險，因為仍是 Merkle–Damgård 結構
安全裕度（Security margin）	較低（256-bit state）	更高（512-bit state）→更高擴散能力與抗差分分析

- Do they provide the same level security?

  SHA-256 與 SHA-512/256 在輸出長度相同的情況下，理論上的安全等級相當（例如對 Collision Attack 都是 $\approx 2^{128}$ 的複雜度），但因為它們的內部結構不同，實際提供的 security margin 不同，所以 SHA-512/256 在某些攻擊模型下安全性略高，以下為幾點 SHA-512/256 的優勢：

  - 更大的 internal state（512 bits）：讓攻擊者更難以構造有效 differential characteristics 。
  - 更長訊息擴展與輪次（80 輪 vs. 64 輪）：每個訊息塊的運算更複雜，對 Collision Attack 與 Preimage Attack 更不友善。
  - 更好的 Avalanche Effect 與訊息擴散性：小小的改變能迅速影響 hash 結果（diffusion 越快，越難預測）。

- Which one is better ?

  SHA-512/256 一般被認為優於 SHA-256。

  原因在於其較大的內部狀態、更強的擴散性，以及在現代 64-bit 處理器上的高效能。雖然兩者輸出長度與理論安全性相同，SHA-512/256 能提供更高的抗結構攻擊能力與安全邊際。

  - 結構設計的優勢
    - SHA-512/256 採用 SHA-512 的核心結構，其基礎是 64-bit 運算架構，相比 SHA-256 的 32-bit 結構，在現代 64-bit 處理器（如 x86-64、ARMv8）上可大幅提升效能。
    - 它將 SHA-512 的輸出從 512 bits 截斷為 256 bits，但保留了完整的 80 輪壓縮函數運算、1024-bit 區塊處理能力與更大的工作變數，這使得它在資訊擴散性與安全性方面更為強大。
    - 較大的 message schedule 與常數表設計（64 個 64-bit 常數）能減少碰撞攻擊中利用結構性弱點（如訊息擴展偏差）的可能。
  - 安全性的比較
    - Collision Resistance（碰撞抵抗）：
      雖然兩者的輸出長度都是 256 bits，因此理論上都提供 2^{128} 的碰撞安全性，但 SHA-512/256 的 internal state 長達 512 bits，意即在 hash 計算過程中，攻擊者需要追蹤的狀態空間

更大，也更難導入有效的 differential characteristics ，因此更能抵抗結構性攻擊。

- ✧ Preimage Resistance（預映像抵抗）：
  兩者皆為 256-bit 輸出，因此理論上都是 2^{256}，但 SHA-512/256 的更多輪次與擴展可提供額外的 security margin ，具更高容錯性。
- ✧ Avalanche Effect & Diffusion：
  SHA-512/256 在設計上提供更高的訊息擴散能力，一個輸入 bit 的改變將影響更多輸出位元（平均超過一半），這讓攻擊者難以預測輸出變化，對抗統計分析特別有效。雖然 SHA-256 的 Avalanche Effect 也很強，但因資料位元較短，擴散速度相對較慢。

- ♦ 效能比較
  - ✧ 在 64-bit 系統（如 PC、伺服器、智慧手機）：
    SHA-512/256 使用 64-bit 加法與位元運算（XOR、rotate），這些操作在 64-bit CPU 上可以單一指令完成，因此運行效能遠高於 SHA-256（需用兩個 32-bit 指令模擬一個 64-bit 運算），尤其在大量資料（如區塊鏈、數位簽章）處理時更明顯。
  - ✧ 在嵌入式系統或 IoT 裝置（多為 32-bit CPU）：
    SHA-256 由於運算單元與記憶體要求較低，反而更適合使用。
- ♦ 相容性與應用
  - ✧ SHA-512/256 與 SHA-256 都是 NIST FIPS 180-4 認可的演算法，支援度高，在各大加密庫（OpenSSL、BoringSSL、libsodium）與系統標準（TLS 1.2/1.3、Bitcoin、Linux kernel）中都可用。
  - ✧ 若系統允許選擇，並且平台為 64-bit，則 SHA-512/256 是更佳選擇，可達到更強安全性與效能雙贏。

2. Exploring SHA as an Encryption Function:

- Can SHA functions be adapted for symmetric encryption? YES

  傳統的 SHA-2（如 SHA-256）是設計來做為單向雜湊函數（不可逆），不適合直接用來加密。但 SHA-3 採用的是「海綿結構（sponge

construction）」，它具有高度彈性，允許輸出任意長度，這讓它可以被用來：

- 當作密鑰衍生函數（KDF）
- 生成類似 stream cipher 的金鑰流（keystream）
- 在某些設計中支援加密與驗證（如 Keyak、KMAC）

- What makes the sponge construction in SHA3 suitable for encryption?

  SHA-3 的 sponge construction 分成兩個階段：

  1. 吸收（Absorb）：將輸入資料 XOR 進內部狀態中，再進行 permutation。
  2. 擠壓（Squeeze）：從內部狀態不斷輸出資料，直到取得所需長度。

  因此，SHA-3 具有以下特點，使其適合對稱加密：

  - 輸出長度可變（Arbitrary-length output）： SHAKE128/SHAKE256 可產生任意長度輸出，很適合拿來做 keystream，像是 stream cipher。
  - Permutation 可逆（內部是對稱的）：雖然雜湊本身是單向，但 permutation 是可逆操作，保留了對稱加密的結構可能性。
  - 輸出不洩漏狀態（state is hidden）：Sponge 的輸出不會暴露中間狀態，自然抵抗 Length Extension Attack，這對加密是必要條件。
  - 良好的擴散與混淆（diffusion and confusion）：小小的輸入變化會導致輸出大幅改變，讓密文看起來像亂數，難以分析。

Problem 2

- Understanding IC
  - f_i ：字母 i 出現的次數（frequency of letter i）。
  - N：整段文字中的總字母數（不含空白與標點）。
- IC Value Comparison:

English	0.065–0.070	字母出現分佈不平均（E, T, A 最常見） E（13%）、T（9%）、A（8%） → 重複字母機率高 → IC 高

Chinese	0.050–0.060	結構為音節，字母分布較為均勻 但常見字如 *de*, *shi*, *le* 等詞會提高某些字母頻率 → 總體 IC 低於英文但高於亂數
randomly generated text	0.038–0.045	每個字母出現機率相同，完全均勻（≈ 1/26） 幾乎不會重複 → IC 值最低，接近理論隨機分布

- Decryption Challenge: Describe your decryption process and methodology.

  計算 IC 後確認為英文，推測應該是單一代換加密法，最常見的情況即為 Caesar Cipher。

  實作一個簡單的 Caesar 解密器，遍歷所有 25 種位移（shift），解開密文後檢查解密結果中是否包含常見英文單字。

```
    COMMON_WORDS = {"THE", "THERE", "AND", "TO", "OF", "IS",
"IN", "THAT", "IT", "ARE", "FOR", "AS", "WITH"}
```

  統計每個解密結果中這些單字出現的次數，作為該位移結果的可信度評分，並找出得分最高的版本，並使用 NLP 自動辨識英文單字邊界得出可讀性高的原始內容。

```
--- Analyzing the ciphertext ---
Index of Coincidence (IC): 0.06810
Most likely to be English (IC ≈ 0.065 ~ 0.070)

--- Decrypting the ciphertext ---
Most likely Caesar shift: 3
Decrypted text :
there are two ways of constructing a software design one way is to
make it so simple that there are obviously no deficiencies and the
other way is to make it so complicated that there are no obvious
deficiencies the first method is far more difficult
```

Problem 3

1. Key Length Estimation: 透過將密文分組並對每組計算 IC，找出平均 IC 值最接近英文自然語言（約 0.068）時對應的金鑰長度。

```
The most probable key length: 4, IC: 0.06837
```

2. Key Recovery: 將密文根據推測的金鑰長度切成數組，每組視為一段使用相同 Caesar 密碼加密的片段，使用 Chi-Square 找出最符合英文字母分布的位移量，推導出每個位置的金鑰字母。

```
Key: NYCU
```

3. Decrypt the ciphertext: 根據找出的金鑰，對密文逐字還原明文，並自動辨識英文單字邊界得出可讀性高的原始內容。
4. 解密完成後，結果會儲存在 plaintext.txt 檔案中，可確認成功解密出《美國獨立宣言》。