**1. Mở đầu**  
Chúng ta chỉ làm việc với các chữ cái thường từ **a** đến **z**, ánh xạ:  
**chữ cái → số = ord(chữ cái) – 96 ∈ [1…26]**.  
Ta luôn chèn thêm ký tự “**x**” vào cuối thông điệp sao cho độ dài của nó ≡ 0 mod 12.  
Sau đó, chia thông điệp thành các khối 12 ký tự.

**2. Xây dựng ma trận “trit” 6×6 từ 12 chữ cái**  
Mỗi chữ cái (có giá trị số ∈ [1…26]) sẽ được biểu diễn trong hệ cơ số 3 thành ba **trit** (số 0, 1 hoặc 2):

**Giá trị ∈ [1…26] →**  
  q₀ = value // 9  
  q₁ = (value % 9) // 3  
  q₂ = value % 3

Giả sử ta có một khối bản rõ gồm 12 chữ cái:  
**plaintext block = L₀L₁…L₁₁**

– Với các chữ cái **L₀…L₅**: ta điền các trit vào **cột 0…5** của **hàng 0,1,2**:

 Với i = 0…5:  
  blockM[0, i] = q₀(Lᵢ)  
  blockM[1, i] = q₁(Lᵢ)  
  blockM[2, i] = q₂(Lᵢ)

– Với các chữ cái **L₆…L₁₁**: ta điền các trit vào **cột 0…5** của **hàng 3,4,5** theo cách tương tự:

 blockM[3, i] = q₀(L₆₊ᵢ), v.v.

Sau bước này, **blockM** là một ma trận 6×6 gồm các trit.

**3. “Xáo trộn” thông qua hoán vị và phép cộng**  
Có sẵn **5 hoán vị 6×6** cố định (A, B, C, D, E).  
Bạn sử dụng một **khóa** là chuỗi chữ cái **k₀k₁…kₘ₋₁**. Mỗi chữ cái trong khóa sẽ tạo ra:

 keyNum = ord(kⱼ) – 97 ∈ [0…25]

Từ đó tính:

 permuteIndex = (keyNum // 5) % 5 ∈ [0…4]  
 addIndex = keyNum % 5 ∈ [0…4]

Với mỗi chữ cái trong khóa (theo thứ tự từ trái sang phải), thực hiện:

* **blockM ← permute(blockM, permuteIndex)**  
  (áp dụng hoán vị đã chọn lên toàn bộ 36 ô của blockM)
* **blockM ← add(blockM, addIndex) mod 3**,  
   trong đó phép **add** là một trong năm quy tắc cộng sau:

 – addIndex = 0: cộng 1 (mod 3) vào mọi ô có (i + j) % 2 == 0  
 – addIndex = 1: khối 3×3 dưới-phải += khối 3×3 trên-trái  
 – addIndex = 2: khối 3×3 trên-trái += khối 3×3 dưới-phải  
 – addIndex = 3: khối 3×3 dưới-trái += khối 3×3 trên-phải  
 – addIndex = 4: khối 3×3 trên-phải += khối 3×3 dưới-trái

Sau khi áp dụng toàn bộ khóa, ma trận blockM đã bị xáo trộn hoàn toàn.

**4. Trích xuất 12 ký tự mã hóa từ mỗi khối**  
Đọc blockM thành chuỗi mới gồm 12 ký tự:

– Với mỗi hàng i = 0…5, tính:

 num = 9·blockM[i,0] + 3·blockM[i,1] + blockM[i,2]  
 Nếu num == 0 → ký tự “0”, ngược lại → chr(num + 96)

– Lặp lại tương tự với mỗi hàng i = 0…5:

 num = 9·blockM[i,3] + 3·blockM[i,4] + blockM[i,5]  
 → ánh xạ sang “0” hoặc từ a đến z như trên

Ta thu được chuỗi 12 ký tự gọi là **resultLetters** cho khối đó.

**5. Phép hoán vị cột cuối cùng**  
Ghép nối tất cả các khối (12 ký tự/block) lại thành chuỗi dài **R**.

Tiếp tục mã hóa bằng một phép hoán vị cột nhỏ:

* Tính keyNums = [ord(k) – 97 cho từng ký tự khóa]
* Tạo reducedKeyNums: từ trái qua phải, giữ lại mỗi giá trị đầu tiên xuất hiện
* Đặt N = độ dài reducedKeyNums. Tạo N “hộp” trống.
* Phân phối R vào các hộp theo vòng tròn (round-robin):  
   R[0] → hộp 0, R[1] → hộp 1, ..., R[N] → hộp 0, …
* Cuối cùng, xuất kết quả bằng cách nối các hộp theo thứ tự tăng dần của reducedKeyNums
* Kết quả chính là **bản mã cuối cùng**, ví dụ:  
  **cnpiaytjyzggnnn…hjt**

**6. Giải mã = thao tác đảo ngược hoàn toàn**

**A) Hoán vị cột ngược**  
– Tính lại reducedKeyNums từ khóa  
– Tính độ dài từng hộp: nếu độ dài bản mã là L, thì hộp j chứa ⌈(L – j)/N⌉ ký tự  
– Đọc các hộp theo thứ tự tăng dần của reducedKeyNums  
– Ghép lại theo round-robin để tái tạo chuỗi R

**B) Phân tách R thành các khối 12 ký tự, tái tạo blockM**  
Với mỗi khối 12 ký tự C₀…C₁₁:

Với i = 0…5, lấy v = (Cᵢ == '0' ? 0 : ord(Cᵢ) – 96)  
 M[i,0] = v // 9, M[i,1] = (v % 9) // 3, M[i,2] = v % 3

Với i = 6…11, làm tương tự cho các cột 3,4,5 của hàng i – 6

**C) Đảo ngược quá trình xáo trộn theo thứ tự ngược của khóa**  
– Với mỗi ký tự khóa (theo thứ tự từ cuối về đầu):

* a = keyNum % 5 → áp dụng **inverse\_add(M, a)**  
   p = (keyNum // 5) % 5 → áp dụng **inverse\_permute(M, p)**
* - inverse\_add: đảo ngược các phép cộng bằng cách trừ mod 3  
   - inverse\_permute: đặt lại từng phần tử vào đúng vị trí ban đầu

**D) Đọc lại 12 chữ cái ban đầu**  
– Với i = 0…5, tính  
 num = 9·M[i,0] + 3·M[i,1] + M[i,2] → chuyển về chữ cái (hoặc “0”)  
– Lặp lại cho M[i,3…5] để lấy toàn bộ 12 chữ cái của khối bản rõ  
(Lưu ý có thể có các ký tự “x” chèn thêm ở cuối – cần loại bỏ)

**7. Tổng kết**  
Khi bạn đưa bản mã:

* **cnpiaytjyzggnnnktjzcvuzjexxkvnrlfzectovhfswyphjt**
* và khóa: **orygwktcjpb**
* vào quy trình giải mã như mô tả, mọi bước đều được đảo ngược chính xác.

#Flag: byuctf{revisreallythestartingpointformostcategoriesiydk}