Corrigé : ave Cesar (X2008 - MP)

Partie I. Codage de César

Question 1.
Le codage de `'maitrecorbeau'` avec un décalage de 5 donne : `'hvdomzxjmwzvp'`.

Question 2.

```
def codageCesar(t, d):
    tprime = []
    for x in t:
        tprime.append((x - d) % 26)
    return tprime
```

Question 3. Il suffit de remplacer d par $-d$:

```
def decodageCesar(t, d):
    tprime = []
    for x in t:
        tprime.append((x + d) % 26)
    return tprime
```

Question 4.

```
def frequences(tprime):
    occ = [0] * 26
    for x in tprime:
        occ[x] += 1
    return occ
```

Question 5. On commence par écrire une fonction qui, à partir du tableau des fréquences, calcule la clef sachant que la valeur maximale de ce tableau correspond à la lettre `'e'` représentée par l'entier 4 :

```
def clef(occ):
    imax = 0
    for i in range(1, 26):
        if occ[i] > occ[imax]:
            imax = i
    return (4 - imax) % 26
```

La fonction de décodage automatique s'écrit alors :

```
def decodageAuto(tprime):
    d = clef(frequences(tprime))
    return decodageCesar(tprime, d)
```

Partie II. Codage de Vigenère

Question 6. Avec la clef `'jean'`, le texte `'becunfromage'` est codé par : `'kichwjrbvegr'`.

Question 7. Cette fois, le décalage se fait vers la droite d'une valeur égale à `c[i % k]`, où i désigne l'indice de la lettre à coder et k la longueur de la clef c.

```
def codageVigenere(t, c):
    k = len(c)
    tprime = []
    for (i, x) in enumerate(t):
        tprime.append((x + c[i % k]) % 26)
    return tprime
```

Question 8. On utilise les formules suivantes, valables pour $(x,y) \in \mathbb{N}^2 \setminus \{(0,0)\}$:

$$\text{pgcd}(x,0) = x, \qquad \text{pgcd}(0,y) = y, \qquad \text{pgcd}(x,y) = \begin{cases} \text{pgcd}(x, y-x) & \text{si } x \leqslant y \\ \text{pgcd}(x-y, y) & \text{sinon} \end{cases}$$

```
def pgcd(a, b):
    x, y = a, b
    while x > 0 and y > 0:
        if x <= y:
            y -= x
        else:
            x -= y
    if x == 0 :
        return y
    else:
        return x
```

Question 9. Pour calculer ce pgcd, on utilise les formules : $\text{pgcd}(0,b) = b$ et $\text{pgcd}(a,b,c) = \text{pgcd}(\text{pgcd}(a,b),c)$.

```
def pgcdDesDistancesEntreRepetitions(tprime, i):
    d = 0
    for j in range(i+3, len(tprime)-2):
        if tprime[i:i+3] == tprime[j:j+3]:
            d = pgcd(d, j-i)
    return d
```

Question 10. Il faut maintenant appliquer la fonction précédente pour tout $0 \leqslant i < n-5$ et calculer le pgcd de toutes les valeurs obtenues pour espérer obtenir la longueur de la clef :

```
def longueurDeLaClef(tprime):
    d = 0
    for i in range(len(tprime)-5):
        d = pgcd(d, pgcdDesDistancesEntreRepetitions(tprime, i))
    return d
```

Question 11. La fonction `pgcdDesDistancesEntreRepetitions` appelle la fonction `pgcd` au plus une fois pour chacune des valeurs $j \in [\![i+3, n-2[\![$, soit au plus $(n-i-5)$ fois. La fonction `longueurDeLaClef` fait donc appel à la fonction `pgcd` un nombre de fois au plus égal à :

$$\sum_{i=0}^{n-6} (n-i-4) = \frac{(n-4)(n-3)}{2} - 1 \sim \frac{n^2}{2}.$$

La fonction `longueurDeLaClef` est de complexité quadratique.

Question 12. Une fois la longueur k de la clef connue, on sait que toutes les lettres séparées par une distance égale à un multiple de k sont codées par le même décalage. On peut donc appliquer la méthode de calcul des fréquences pour déterminer ce décalage, et donc chacune des lettres de la clef.
Concrètement, pour tout $i \in [\![0, k-1]\!]$, les lettres du sous-tableau $t[i :: k]$ ont toutes été codées avec le même décalage d_i. On peut donc utiliser la fonction `decodageAuto` écrite à la question 5 pour décoder cette partie du message.

```
def decodageVigenereAuto(tprime):
    n = len(tprime)
    k = longueurDeLaClef(tprime)
    t = [None] * n
    for i in range(k):
        t[i::k] = decodageAuto(tprime[i::k])
    return t
```