Notons S1 la solution proposée le 23/02.
3 errata concernant S1 :

1. Question 7 - Ligne 5 : Remplacer $cle := irem(i,k)$; par $cle := c[irem(i,k)]$;
2. Question 13 : Il manque tout le début : $cle := proc(...)...$
   Remplacer toute la question 13 par :

```
antiCle :=proc(t',n,k)    # Si la clé est "jean" ie "9,4,0,13", on cherche antiC="-9,-4,0,-13"
local antiC,i,q,iMot,j,f;
for i from 0 to k-1 do
      q :=floor((n-1-i)/k);
      iMot :=array(0..q-1);              # iMot est le sous-tableau t'[i],t'[i+k],t'[i+2k],...
      for j from 0 to q-1 do
            iMot[j] ;=t'[i+k*j]
      od;
      f :=frequences(iMot,q)
      antiC[i] :=-imageDuE(f)
od;
return antiC;
end;


decodageVigenereAuto :=proc(t',n)
local k,c;
k :=longueurDeLaCle(t',n);
c :=antiCle(t',n,k);
return codageVigenere(t',n,c,k)
end;
```

3. Dans tout le problème, on a besoin de connaître des entiers *modulo* 26 pour les identifier à une lettre de l'alphabet.
   Si $p \in \mathbb{N}$, alors irem(p,26) donne ce résultat ; si $p < 0$, irem(p,26) donne un résultat ... mais pas le bon.
   Une solution est de remplacer tous les "irem(x,26)" par "x *mod* 26" ; c'est simple, mais *mod* est une function Maple non-élémentaire donc je pense que ce n'est pas souhaitable.
   Une meilleure solution est de remplacer tous les "irem(x,26)" par "x-26*floor(x/26)" (floor est pour l'X une instruction élémentaire), ou plutôt d'ajouter au début de la question 1 :

```
reste :=proc(x)
return(x-26*floor(x/26))
end;
```

   et de remplacer tous les "irem(x,26)" par "reste(x)"

Quelques remarques pour tout le problème :

- On doit manipuler des "tableaux", pas des "list" ; tous les [seq()], op, nops ... sont des particularités Maple non admises.
  Un tableau se déclare par "a :=array(-2..56)" et ses éléments sont appelés par "a[-2],...,a[56]".
  En 2008, les tableaux devaient être indexés A PARTIR DE 0 : il faut respecter cette consigne. Certaines années, on indexe à partir de 1 ... ; il faut lire attentivement le texte !
  Selon un rapport, "a :=array(-2..56,0)" pour créer un tableau d'éléments nuls est trop "Maple" donc moins bien que
  a :=array(-2..56) ;

for i from -2 to 56 do
    a[i] :=0
od ;
... mais c'est toléré, et bien pratique, donc je vous le conseille.

- Attention aux " ;" oubliés (erreur grave) et aux " ;" en trop (c'est aussi grave !) : pas de " ;" après un do, un then ...Et " :" a le même effet que " ;" (sauf l'affichage mais ce n'est pas le sujet)
- On ne doit pas écrire 2 instructions sur la même ligne ; cela ne choque pas les profs-de-math, mais ça choque les informaticiens, donc ici c'est grave.
  N'hésitez pas à "tabuler" énergiquement pour écrire l'intérieur d'une boucle ou un test. Ce ne sera jamais trop clair.
- Attention aux initialisations mal placées (dans la boucle ou avant la boucle), aux bornes erronées.
  Pensez à utiliser les "procedure" déjà écrites (ou supposées écrites) ; il n'est pas interdit de scinder une question en 2 "procedure" (ex "reste", "imageDuE", "antiCle").

Question par question :

3 - C'est mieux d'utiliser Q2 au lieu de recommencer.

4 - C'est maladroit de lire 26 fois le mot pour trouver les 26 fréquences (solutions avec 2 boucles imbriquées)

8 - Le plus court (le mieux) est :

```
pgcd :=proc(a,b)        #Solution S2
if b=0 then
    return a
else
    return pgcd(b,irem(a,b))
fi ;
end ;
```

Il est superflu de distinguer les cas $a < b$ et $b < a$.
Dans S1, vous avez une solution plus "machine" qui n'utilise pas explicitement la division euclidienne (irem) mais qui fonctionne pareil :
Pour S1, pgcd(24,9)=pgcd(15,9)=pgcd(6,9)=pgcd(6,3)=pgcd(3,3)=pgcd(3,0)=3.
Pour S2, pgcd(24,9)=pgcd(9,6)=pgcd(6,3)=pgcd(3,0)=3.

9 - Il n'est pas utile de conserver les indices correspondant aux répétitions trouvées dans un tableau.

10 - On ne demande pas le plus petit pgcd, mais le pgcd des pgcd.

13 - On se ramène à un codage/décodage déjà vu, soit à codageVigenere en prenant la clé inverse (comme pour le decodageCesar), soit on utilise decodageCesar sur chaque sous-mot iMot.