Langage de programmation : Maple

1) En utilisant un décalage de 5, le codage du texte maitrecorbeau donne h v d o m z x j m w z v p .

2)
```
codageCesar := proc(t,n,d)
local tcode,i;
tcode := array(0..n-1);
for i from 0 to n-1 do
        tcode[i] := irem(t[i]-d, 26)
od;
return tcode;
end;
```

3)
```
decodageCesar := proc(t,n,d)
local tdecode;
tdecode := codageCesar(t,n,26-d);
return tdecode;
end;
```

4)
```
frequences := proc(t',n)
local freq, i, lettre;
freq := array(0..25)
for i from 0 to 25 do
        freq[i] := 0
od;
for i from 0 to n-1 do
        lettre := t'[i];
        freq[lettre] := freq[lettre] + 1
od;
return freq;
end;
```

5)
```
imageDuE := proc (f)
local fmax, image, i;
fmax := f[0];
image := 0;
for i from 1 to 25 do
        if  f[i] > fmax  then
                fmax := f[i];
                image := i
        fi;
od;
return image;
end;
```

```
decodageAuto := proc (t', n)
local f, image, d;
f := frequences (t', n);
image := imageDuE (f);
d := irem (4 - image, 26);
return decodageCesar (t', n, d);
end;
```

6) En utilisant la clé "jean", le codage du texte becunfromage donne :

k i c h w f j n b v e g z

7)
```
codageVigenere := proc (t, n, c, k)
local tcode, i, cle;
tcode := array (0 .. n-1);
for i from 0 to n-1 do
        cle := c[irem (i, k)];
        tcode[i] := irem (t[i] + cle, 26)
od;
return tcode;
end;
```

8)
```
pgcd := proc (a,b)
if a = 0 then return b else
      if b = 0 then return a else
            if a > b then return pgcd (a-b,b) else
                  return pgcd (a, b-a);
            fi;
      fi;
fi;
end;
```

9)
```
pgcdDesDistancesEntreRepetitions := proc (t',n,i)
local seq1, seq2, seq3, pDDER, j;
seq1 := t'[i];
seq2 := t'[i+1];
seq3 := t'[i+2];
pDDER := 0;
for j from i+3 to n-3 do
      if t'[j] = seq1 and t'[j+1] = seq2 and t'[j+2] = seq3 then
            pDDER := pgcd (pDDER, j-i)
      fi;
od;
return pDDER;
end;
```

10)
```
longueurDeLaClé := proc (t',n)
local longueur, i;
longueur := 0;
for i from 0 to n-6 do
      longueur := pgcd ( longueur, pgcdDesDistancesEntreRepetitions (t',n,i))
od;
return longueur;
end;
```

11) pgcd Des Distances Entre Repetitions (t',n,i) appelle au plus $(n-3)-(i+3)+1$ fois la fonction pgcd, soit $n-i-5$ fois.
longueurDeLaCle (t',n) appelle donc au plus:

$$\sum_{i=0}^{n-6} 1 + (n-i-5) \quad \text{fois la fonction pgcd}$$

soit $$\sum_{i=0}^{n-6} (n-i-4) = \sum_{k=2}^{n-4} k = \frac{1}{2}\left[(n-4+2)(n-4-1)\right] = \frac{1}{2}(n-2)(n-5)$$
$$= O(n^2)$$

12) t' contient le texte codé de longueur n.
longueurDeLaCle (t',n) donne un nombre k. on suppose $k \geq 1$
$t'' = [\, t'[0], t'[k], \ldots, t'[k\,p\,]\,]$ est un texte codé de longueur p $(p = E(\frac{n}{k}))$
frequences (t'', p) donne le tableau des frequences de 0...25 dans t''
On peut admettre que le nombre le plus frequent code la lettre e.
imageDuE permet de dire que, dans t'', e est codé par image, donc a est codé par image − 4 qui est la premiere lettre de la clé.
On recommence avec $t'' = [\, t'[1], t'[k+1], \ldots, t'[p\,k+1]\,]$ pour la seconde lettre, etc. –

13)
```
decodageVigenereAuto := proc (t',n)
local k, c, tdecode, i;
k := longueurDeLaCle (t',n);
c := cle (t',n,k);
tdecode := array (0..n-1);
for i from 0 to n-1 do
        d := c [irem (i,k)];
        tdecode [i] := irem ( t[i] + d , 26)
od;
return tdecode;
end;
```