X 2009 : Info pour non – informaticiens

```
> N := 14; NpN := N^N;    a := Array(0..N-1);
```

$$N := 14$$

$$NpN := 11112006825558016$$

$$a := \begin{bmatrix} 0 .. 13\ Array \\ Data\ Type: anything \\ Storage: rectangular \\ Order: Fortran\_order \end{bmatrix} \quad (1)$$

```
> ArrayNumElems(a);
```

$$14 \quad (2)$$

```
> AAL c'est AfficheArraycommeuneListe  pour n'avoir que les nombres et les ,
> AAL :=proc(a) convert(a, list) end;
```

$$AAL := \mathbf{proc}(a)\ convert(a, list)\ \mathbf{end\ proc} \quad (3)$$

Question 1

```
> DecomposerBase :=proc(N, k)
      local a, boulot; a := array(0..N-1);
      boulot :=proc(m, rang)
        if m = 0
        then NULL
        else a[rang] := irem(m, N); boulot(iquo(m, N), rang + 1)
        fi;
      end;
    boulot(k, 0); a end:
> iquo(843, 14); irem(843, 14);
```

$$60$$

$$3 \quad (1.1)$$

```
> AAL(DecomposerBase(14, 843));
```

$$[3, 4, 4, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}] \quad (1.2)$$

Question 2

```
> DecomposerFact :=proc(N, k)
      local a, boulot, i;
      a := Array(0..N-1);
      for i from 0 to (N-1) do a[i] := 0 od;
      boulot :=proc(m, rang, base)
        if m ≤ 0
```

```
            then NULL
            else a[rang] := iquo(irem(m, base*(rang + 1)), base);
                 boulot(m - a[rang]*base, rang + 1, base*(rang + 1))
            fi;
        end;
    boulot(k, 1, 1); evala(a) end:
> AAL(DecomposerFact(4, 17));
```

$$[0, 1, 2, 2] \qquad \textbf{(2.1)}$$

> On a en effet 17 = 2*3! + 2*2! + 1*1! + 0

```
> AAL(DecomposerFact(10, 171718));
```

$$[0, 0, 2, 3, 4, 2, 0, 2, 4, 0] \qquad \textbf{(2.2)}$$

> 0+0*1!+2*2!+3*3!+4*4!+2*5!+2*7!+4*8!; 171718
Commentaire personnel : j'ai rarement fait autant d'erreurs pour `écrire` une fonction aussi simple

Question 3

```
> RetirerListe := proc(L, l, j) [op(L[1..j-1]), op(L[j+1..l])] end:

  RetirerListe([$1..23], 23, 14);
```

$$[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23] \qquad \textbf{(3.1)}$$

Il y a même encore plus simple : on n'a pas besoin de l, en Maple le dernier d 'une liste se désigne par l'indice -1

Il est même possible qu
'il y ait une fonction Maple qui fasse directement ce qui précède

> Ceci dit il est bien évident que ce n'est pas cela que l'auteur de l'énoncé avait en tête, alors je recommence avec des arrays

```
> Retirer := proc(L, l, j) local new, k; new := Array(0..l-1);
            for k from 0 to (j-1) do new[k] := L[k] od;
            for k from j to l - 1 do new[k] := L[k+1] od; new end:

  test := array(0..9, [1, 5, 9, 7, 5, 3, 2, 4, 8, 6]):
  AAL(Retirer(test, 9, 2));
```

Question 4

```
> EcrirePermutation := proc(N, k) local a, σ, L, p, i;
      σ := Array(0..N-1);
    L := Array(0..N-1);for i from 0 to N-1 do L[i] := i od;
    a := DecomposerFact(N, k);
     for p from (N-1) by (-1) to 0 do σ[(N-1)-p] := L[a[p]]; L := Retirer(L, p, a[p])
```

```
        od;
      σ
    end:;
> AAL(EcrirePermutation(10, 20813));
                    [0, 1, 6, 2, 9, 5, 3, 8, 7, 4]                    (4.1)
```

Question 5

```
>
```

piège !

```
> Chiffrer := proc(N, k, b) local σ, c, p;
    σ := EcrirePermutation(N, k); c := Array(0..nops(AAL(b)) − 1); for p from 0 to (N
      − 1) do c[p] := b[σ[p]] od; c end:

> AAL(test);
                    [1, 5, 9, 7, 5, 3, 2, 4, 8, 6]                    (5.1)

> AAL(Chiffrer(10, 37813, test));
                    [1, 5, 6, 3, 2, 9, 4, 7, 8, 5]                    (5.2)

> inverser := proc(perm) local mrep, k, L; L := nops(AAL(perm)); mrep := Array(0..(L
      − 1));
    for k from 0 to (L − 1) do mrep[perm[k]] := k od; mrep end:
>
```

pour tester

```
> Dechiffrer := proc(N, k, b) local σ, c, p;
    σ := inverser(EcrirePermutation(N, k)); c := Array(0..nops(AAL(b)) − 1); for p
      from 0 to (N − 1) do c[p] := b[σ[p]] od; c end:
> AAL(Dechiffrer(10, 37813, Chiffrer(10, 37813, test)));
                    [1, 5, 9, 7, 5, 3, 2, 4, 8, 6]                    (5.3)
```

Question 6

```
> N := 2^64; n := 2^32;

                    N := 18446744073709551616
                    n := 4294967296                                   (6.1)
```

autour de xor

> la fonction Xor est "bien sûr" toute prête dands Maple, comme cela ne fait pas du tout partie de ce que vous avez à connaitre, je l'ai re-fabriquée ci-dessous. Le monXor qui suit n'a aucune prétention de rapidité ou autre. Par la suite j'utiliserai le Xor tout prêt du package Bits

```
> convert(842, base, 2); convert(333, base, 2);
```

$$[0, 1, 0, 1, 0, 0, 1, 0, 1, 1]$$

$$[1, 0, 1, 1, 0, 0, 1, 0, 1] \quad \textbf{(6.1.1)}$$

```
> monXor := proc(n1, n2) local ln1, ln2, lxor, res, petit, grand, debut, fin, somme2, copie,
      long;
       ln1 := nops(n1); ln2 := nops(n2); petit := min(ln1, ln2); grand := max(ln1,
      ln2);
       res := [$1..grand]; debut := [$1..petit]; fin := [$(petit + 1)..grand];
       somme2 := proc(k, x, y) res[k] := (x + y) mod 2 end;
       copie := proc(k, m) res[k] := m[k] end;
       map(k→somme2(k, n1[k], n2[k]), debut);
       if ln1 < ln2 then long := n2 else long := n1 fi;
       map(k→copie(k, long), fin); res
    end:
```

```
> monXor([0, 1, 0, 1, 0, 0, 1, 0, 1, 1], [1, 0, 1, 1, 0, 0, 1, 0, 1]);
```

$$[1, 1, 1, 0, 0, 0, 0, 0, 0, 1] \quad \textbf{(6.1.2)}$$

```
> with(Bits) :
> Xor(842, 333);
```

$$519 \quad \textbf{(6.1.3)}$$

```
> Join(monXor([0, 1, 0, 1, 0, 0, 1, 0, 1, 1], [1, 0, 1, 1, 0, 0, 1, 0, 1]));
```

$$519 \quad \textbf{(6.1.4)}$$

```
>
```

autour de F

> Je choisis comme 'fouillis F' : je multiplie k par r modulo n

```
> F := proc(k, r) (k·r) mod n end;
```

$$F := \textbf{proc}(k, r)\ mod(k * r, n)\ \textbf{end proc} \quad \textbf{(6.2.1)}$$

```
> F(123456, 147258);
```

$$1000014464 \quad \textbf{(6.2.2)}$$

```
> FeistelTour := proc(k, b) local q, r; r := b mod n; q := (b − r)/n; n·r + Xor(q, F(k, r))
    end:
```

Question 7

```
> FeistelInverseTour := proc(k, b) local q, r; r := b mod n; q := (b − r)/n; n·Xor(r, F(k,
      q)) + q end;
```

$$FeistelInverseTour := \textbf{proc}(k, b)\ \textbf{local}\ q, r;\ r := mod(b, n);\ q := (b - r)/n;\ n * Bits:\text{-}Xor(r, F(k, q)) + q\ \textbf{end proc} \quad \textbf{(7.1)}$$

```
> FeistelTour(369852, 123456);
```

$$530242193270528 \quad \textbf{(7.2)}$$

```
> FeistelInverseTour(369852, 530242193270528);
```

123456 **(7.3)**

Question 8

autour de K

> Je limite la taille de K à 50, et je ne prend pas la peine de vérifier que les 50 termes sont distincts

```
> monK := map(k→rand( ), [$0..50]) :
```

```
> Feistel :=proc(K, l, b) local k, c; c := b; for k from 1 to (l − 1) do c
      := FeistelTour(K[k], c) od; c end:
```

```
> Feistel(monK, 18, 123456);
```

12662283220007654592 **(8.1)**

Question 9

```
> FeistelInverse :=proc(K, l, b) local k, c; c := b; for k from (l − 1) by (−1) to 1 do c
      := FeistelInverseTour(K[k], c) od; c end:
```

```
> FeistelInverse(monK, 18, 12662283220007654592);
```

123456 **(9.1)**

```
>
```

Question 10

> Alors que depuis le début le problème traitait d'entiers, voilà qu'on est censé passer aux bits! Il y a pour cela la fonction de Q1 que l'on avait eu pour ordre d'écrire ... à l'envers

autour de Sigma

```
> Σ :=proc(x) Feistel(monK, 18, x + 1) end;
```

$\Sigma := \mathbf{proc}(x)\ Feistel(monK, 18, x+1)\ \mathbf{end\ proc}$ **(10.1.1)**

> EBC pour EcritureBinaireComplete

```
> EBC :=proc(x) local bimales, boulot; bimales := Array(0..63, 0);
      boulot :=proc(k, y)
          if k = 64
          then NULL
          else bimales[63 − k] := (y mod 2);
              boulot(k + 1, iquo(y, 2)) fi end;
      boulot(0, x); bimales end:
```

```
> AAL(EBC(2^3 + 2^17 + 2^51 + 2^52));
```

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0] **(10.1)**

```
> Sequence :=proc(n) local gros, EcritDansGros, p; gros := Array(0..(n − 1), 0);
```

```
EcritDansGros := proc(k) local Acopier, i;
    Acopier := EBC(Σ(k));
    for i from 0 to 63 do gros[k·64 + i] := Acopier[i] od end;
for p from 0 to (n/64 - 1) do EcritDansGros(p) od;
gros end:
```

```
> AAL(Sequence(256));
```

[1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1,
1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1,
0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
1, 0, 0, 0, 1, 0, 0, 1, 0, 0] **(10.2)**

```
>
```

Question 11

```
> V1 := proc(liste01) local n0, n1, long; long := nops(liste01); n1 := convert(liste01, `+`);
      (long - 2·n1)^2 / long ·1. end;
```

V1 := proc(liste01) **(11.1)**
local n0, n1, long;
long := nops(liste01); n1 := convert(liste01, `+`); (long − 2 * n1)^2 * 1. / long
end proc

```
> V1([0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1]);
```

0.3103448276 **(11.2)**

```
> CalculerV1 := proc(n) V1(AAL(Sequence(n))) end;
```

CalculerV1 := proc(n) V1(AAL(Sequence(n))) end proc **(11.3)**

```
> CalculerV1(640);
```

12.65625000 **(11.4)**

Question 12

```
> V2 := proc(liste01) local n0, n1, n00, n01, n10, n11, long, enregistre, k;
      long := nops(liste01); n0 := 0; n1 := 0; n00 := 0; n01 := 0; n10 := 0; n11 := 0;
      enregistre := proc(a, b)
          if a = 0 then n0 := n0 + 1 else n1 := n1 + 1 fi;
          if    (a = 0 and b = 0) then n00 := n00 + 1
          elif  (a = 1 and b = 0) then n10 := n10 + 1
          elif  (a = 0 and b = 1) then n01 := n01 + 1
          else  n11 := n11 + 1 fi end;
```

```
        for k from 1 to long − 1 do enregistre(liste01[k], liste01[k + 1]) od;
        if liste01[long] = 0 then n0 := n0 + 1 else n1 := n1 + 1 fi;
        4/(long − 1) · (n00^2 + n10^2 + n01^2 + n11^2) - (2/long)(n0^2 + n1^2) + 1.
    end:
```

```
> CalculerV2 := proc(n) V2(AAL(Sequence(n))) end;
```

$$CalculerV2 := \mathbf{proc}(n)\ V2(AAL(Sequence(n)))\ \mathbf{end\ proc} \qquad \textbf{(12.1)}$$

```
> CalculerV2(640);
```

$$666.3302083 \qquad \textbf{(12.2)}$$

```
>
```