1.
   ```
   DecomposerBase :=proc(N,k)
   local autreK,tab ;i ;
   autreK :=k ;
   tab :=array(0..N-1) ;
   for i from 0 to N-1 do
           tab[i] :=rem(autreK,N) ;
           autreK :=quo(autreK,N)
   od ;
   return tab ;
   end ;
   ```

2.
   ```
   DecomposerFact :=proc(N,k)
   local autreK,tab ;i ;
   autreK :=k ;
   tab :=array(0..N-1) ;
   for i from 0 to N-1 do
           tab[i] :=rem(autreK,i+1) ;
           autreK :=quo(autreK,i+1)
   od ;
   return tab ;
   end ;
   ```

3.
   ```
   Retirer :=proc(L,l,j)
   local autreL,i ;
   autreL :=array(0..l-2) ;
   for i from 0 to l-2 do
           if i<l then
                   autreL[i] :=L[i]
           else
                   autreL[i] :=L[i+1]
           fi ;
   od ;
   return autreL ;
   end ;
   ```

4.
   ```
   EcrirePermutation :=proc(N,k)
   local L,i,kDecomp,LPermute,a ;
   L :=array(0..N-1)
   for i from 0 to N-1 do
           L[i] ;=i
   od ;
   kDecomp :=DecomposerFact(N,k) ;
   LPermute :=array(0..N-1) ;
   for i from 0 to N-1 do
           a :=kDecomp[N-1-i] ;
           LPermute[i] :=L[a] ;
           L :=Retirer(L,N-i,a)
   od ;
   return LPermute ;
   end ;
   ```

5.
   ```
   Chiffrer :=proc(N,K,b)
   local LPermute ;
   LPermute :=EcrirePermutation(N,k) ;
   return LPermute(b) ;
   end ;
   ```

```
DeChiffrer :=proc(N,K,b)
local LPermute,i;
LPermute :=EcrirePermutation(N,k);
i :=0;
while b<>LPermute[i] do
        i :=i+1
od;
return i;
end;
```

6. FeistelTour :=proc(k,b)

   ```
   FeistelTour :=proc(k,b)
   local q,r;
   q :=rem(b,2^32);
   r :=xor(quo(b,2^32),F(k,q));
   return q*2^32+r;
   end;
   ```

7. FeistelInverseTour :=proc(k,b)

   ```
   FeistelInverseTour :=proc(k,b)
   local q,r,rFeistel;
   rFeistel :=rem(b,2^32);
   r :=quo(b,2^32);
   q :=xor(rFeistel,F(k,r));
   return q*2^32+r;
   end;
   ```

8. Feistel :=proc(K,l,b)

   ```
   Feistel :=proc(K,l,b)
   local bloc,i;
   bloc :=b;
   for i from 0 to l-1 do
           bloc :=FeistelTour(K[i],bloc)
   od;
   return bloc;
   end;
   ```

9. FeistelInverse :=proc(K,l,b)

   ```
   FeistelInverse :=proc(K,l,b)
   local bloc,i;
   bloc :=b;
   for i from 0 to l-1 do
           bloc :=FeistelInverseTour(K[l-1-i],bloc)
   od;
   return bloc;
   end;
   ```

10. Sequence :=proc(n)

    ```
    Sequence :=proc(n)
    local nbre,seq,i,j,ichiffre;
    nbre :=quo(n,64);
    seq :=array(0..n);
    for i from 0 to nbre-1 do
            ichiffre :=Sigma(i);
            for j from 0 to 63 do
                    seq(64*(i+1)-1-j) :=rem(ichiffre,2);
                    ichiffre :=quo(ichiffre,2)
            od;
    od;
    return seq;
    end;
    ```

11.
```
CalculerV1 :=proc(n)
local seq,n1,n0,i ;
seq :=Sequence(n) ;
n1 :=0 ;
for i from 0 to n-1 do
        n1 :=n1+seq(i)
od ;
n0 :=n-n1 ;
return $(n0 - n1)^2$/n ;
end ;
```

12.
```
CalculerV2 :=proc(n)
local seq,n1,n0,n01,n00,n01,n11,i ;
seq :=Sequence(n) ;
n1 :=0 ;
n01 :=0 ;
n11 :=0 ;
for i from 0 to n-2 do
        if seq[i]=0 then
                n01 :=n01+seq[i+1]
        else
                n11 :=n11+seq[i+1] ;
                n1 :=n1+1 ;
        fi ;
od ;
n10 :=n1-n11 ;
n00 :=n-1-n1-n01 ;
n1 :=n1+seq[n-1] ;
n0 :=n-n1 ;
return $(n00^2 + n01^2 + n10^2 + n11^2)$*4/(n-1)-$(n0^2 + n1^2)$*2/n+1 ;
end ;
```