EPREUVE INFORMATIQUE COMMUNE X MP-PC

On utilise une fonction auxiliaire taille convenant aussi bien pour les listes que pour les tableaux. Le tableau n'est pas le type le plus simple à utiliser en Maple.

```
est_permutation:=proc(t) local n,u,nbr_images,i;
> n:=taille(t);
  u:=Array(1..n);
  nbr_images:=0;
> for i from 1 to n do u[i]:=0; od;
> for i from 1 to n do if (t[i]<=n) and u[t[i]]=0 then
                  nbr_images:=nbr_images+1;u[t[i]]:=1:fi;od;

> return(evalb(nbr_images=n))
> end:
>
> taille:=proc(a);if type(a,list) then nops(a) elif type(a,Array)
  then nops(op(3,a)) fi;;end:

>
```

Question 3

On suppose les permutations de même taille

```
> composer:=proc(t,u) local n,v,i;
> n:=taille(u);
> v:=Array(1..n);
> for i from 1 to n do v[i]:=t[u[i]];od;
> return(v);end:
```

Question 4: on utilise la "définition" de l'énoncé

```
> inverser:=proc(t) local n,v,i ;
> n:=taille(t);
  v:=Array(1..n);
> for i from 1 to n do v[t[i]]:=i;od;
> return(v);end:
>
```

Question 4: id et [n,1,2,...,n-1]

Question 5: on écrit d'abord une procédure est_identité (dont les sens est clair)

```
>  est_identité:=proc(v) local n,i;
  n:=taille(v);
> i:=1; while (i<=n) and v[i]=i do i:=i+1;od;
> if i=n+1 then return(true) else return(false);fi;end:
> est_identité([2,1,3]);
>
```

false

```
> ordre:=proc(t) local v, ord;
> v:=t;ord:=1;
> while not est_identité(v) do v:=composer(v,t);ord:=ord+1;od;
  return(ord);end:
```

Partie II Manipuler les permutations

Question 6

```
> période:=proc(t,i) local n,k,j;
  n:=taille(t);
  k:=1;
  j:=t[i];
> while j<>i do k:=k+1; j:=t[j];od;
  return(k);
> end;
```

Question 7

On calcule les images successives de i jusqu'à obtenir j ou i

```
>
```

```
> estDansOrbite:=proc(t,i,j) local k;
  if i=j then return(true) else
>      k:=t[i];while (k<>j) and (k<>i) do k:=t[k];od;
     if k=j then return(true) else return (false);fi;fi;
> end;
```

Question 8

Une permutation s est une transposition si et seulement si elle a exactement deux éléments non invariants.On compte ce nombre dans la variable s

```
> estTransposition:=proc(t)local n,s,i;
>   n:=taille(t);s:=0;
> for i from 1 to n do if t[i]<>i then s:=s+1;fi;od;
  if s=2 then return(true) else return(false);fi;end;
```

Question 9

Une permutation est un cycle si et seulement si elle possède une seule orbite de plus d'un élément. On commence par chercher les points fixes comptés dans la variable s et on détecte s'il y a une autre orbite. on compte alors le nombre d'éléments de cette orbite .

```
> estCycle:=proc(t) local s,n,k,i,j,m;
>   s:=0;n:=taille (t);k:=0;
> for i from 1 to n do if t[i]=i then s:=s+1 else k:=i;fi;od;
  j:=t[k];m:=2;while t[j]<>k do j:=t[j];m:=m+1 od;
  if m+s=n then return(true) else return(false);fi;
> end:
```

Question 10
On calcule la préride d'un élément qui également celle de tous les éléments de l'orbite. On crée un tableau P de périodes initialisé à 0 et on ne traite que les éléments tels que P[i]=0.La complexité amortie est bien linéaire.

```
> periodes:=proc(t) local n,P,i,j,k,s;
  n:=taille(t);
> P:=Array(1..n);
> for i from 1 to n do P[i]:=0;od;
> for j from 1 to n do if P[j]=0 then
>                   k:=j;s:=1;while t[k]<>j do s:=s+1;k:=t[k] od;
                    k:=j;P[k]:=s; while t[k]<>j do
  k:=t[k];P[k]:=s;od;fi;
  od;return(P);end:
>
```

Question 11

```
> itererEfficace:=proc(t,k)local P,n,u,i,nbit,elem,j;
> P:=periodes(t);
  n:=taille(t); u:=Array(1..n);
> for i from 1 to n do
       nbit:=irem(k,P[i]);elem:=i;for j from 1 to nbit do
  elem:=t[elem] od;
  u[i]:=elem; od;return(u);end:
>
```

Question 12
[3,3,1,5,4], composée de deux cycles disjoints d'ordres respectifs 2 et 3 est d'ordre 6
Question 13
(On rappelle que le coût est logarithmique en a et b, donc bilinéaire par rapport aux tailles machine de a et b) Pour la récursivité, on a besoin d'envisager le cas où b=0

```
> pgcd:=proc(a,b)
  if b=0 then return(a) else return pgcd(b, irem(a,b));fi;
> end:
> ppcm:=proc(a,b);
  return(a*b)/pgcd(a,b);end:
>
```

Question 15. La notion de PPCM de plusieurs entiers repose sur l'associativité du PPCM.
On calcule de proche en proche (avec la variablr ord) le PPCM des ordres de tous les éléments à partir du tableau périodes en utilisant l'associativité et en utilisant une seule fois chaque ordre différent de 1. Chaque période d'un élément est comprise entre 1 et n;. Il suffit donc d'utiliser un tableau per de booleens indexés entre 1 et n, avec per[i]=true si et seulement si i est période différente de 1 d'un élément de [1,n] pour t

```
>
```

```
> ordreEfficace:=proc(t)local n, P,per,i ,ord;
> n:=taille(t);P:=periodes(t);
> per:=Array(1..n);
  for i from 1 to n do per[i]:=false;od;
> for i from 1 to n do if P[i]>1 then per[P[i]]:=true fi; od;
> ord:=1; for i from 1 to n do if per[i] then ord:=ppcm(ord,i)
  fi;od;
> return(ord);end:

>

> 
```