1.
```
estPermutation :=proc(t)
local n,vu,i;
n :=taille(t);
vu :=allouer(n);
for i from 1 to n do
        vu[i] :=0
od;
for i from 1 to n do
        if t[i]<0 or t[i]>n then
                return faux
        else
                vu[t[i]] :=1
        fi
od;
for i from 1 to n do
        if vu[i]=0 then
                return faux
        fi;
od;
return vrai;
end;
```
2.
```
composer :=proc(t,u)
local n,comp,i;
n :=taille(t);
comp :=allouer(n);
for i from 1 to n do
        comp[i] :=u[t[i]]
od;
return comp;
end;
```
3.
```
inverser :=proc(t)
local n,inv,i;
n :=taille(t);
inv :=allouer(n);
for i from 1 to n do
        inv[t[i]] :=i
od;
return inv;
end;
```
4. Id est d'ordre 1 et le cycle (1,2,...,n) est d'ordre n.
5.
```
test :=proc(t)
local n,i;
n :=taille(t);
for i from 1 to n do
        if t[i]<>i then
                return faux
        fi;
od;
return vrai;
end;
```

```
ordre :=proc(t)
local n,itere,i,ord;
n :=taille(t);
itere :=allouer(n);
for i from 1 to n do
        itere[i] :=t[i]
od;
ord :=1;
while test(itere)=faux do
        itere :=composer(itere,t);
        ord :=ord+1
od;
return ord;
end;
```

6.
```
periode :=proc(t,i)
local period,itere;
period :=1;
itere :=t[i];
while itere<>i do
        itere :=t[itere];
        period :=period+1
od;
return period;
end;
```

7.
```
estDansOrbite :=proc(t,i,j)
local period,itere,k;
period :=periode(t,i);
itere :=i;
for k from 1 to period do
        if itere=j then
                return vrai
        else
                itere :=t[itere];
        fi;
od;
return faux;
end;
```

8.
```
estTransposition :=proc(t)
local n,CardSupport,i;
n :=taille(t);
CardSupport :=0;
for i from 1 to n do
        if t[i]<>i then
                CardSupport :=CardSupport+1;
        fi;
od;
if CardSupport=2 then
        return vrai
else
        return faux
fi;
end;
```

9.
```
estCycle :=proc(t)
local n,CardSupport,iDansSupport,i ;
n :=taille(t) ;
CardSupport :=0 ;
iDansSupport :=0 ;
for i from 1 to n do
        if t[i]<>i then
                CardSupport :=CardSupport+1 ;
                iDansSupport :=i
        fi ;
od ;
if iDansSupport>0 and CardSupport=periode(t,iDansSupport) then
        return vrai
else
        return faux
fi ;
end ;
```

10.
```
periodes :=proc(t)
local n,leader,i,period ;
n :=taille(t) ;
leader :=allouer(n) ;                   (leader[i] sera le plus petit élément de l'orbite de i)
for i from 1 to n do
        leader[i] :=0
od ;
period :=allouer(n) ;
for i from 1 to n do
        if leader[i]=0 then                   (dans ce cas, on n'a pas encore rencontré l'orbite de i)
                leader[i] :=i ;               (car i est alors le plus petit élément de son orbite)
                itere :=t[i] ;
                period[i] :=1 ;
                while itere<>i do                  (on parcourt l'orbite de i)
                        leader[itere] :=i ;
                        period[i] :=period[i]+1 ;
                        itere :=t[itere]
                od ;
        fi ;
od ;
for i from 1 to n do
        period[i] :=period[leader[i]]          (tous les éléments de l'orbite de i ont la même période)
od ;
return period ;
end ;
```

11.
```
itereEfficace :=proc(t,k)
local n,itere,perio,i,j,m ;
n :=taille(t) ;
itere :=allouer(n) ;
perio :=periodes(t) ;
for i from 1 to n do
        j :=reste(k,perio[i]) ;
        itere[i] :=i ;
        for m from 1 to j do
                itere[i] :=t[itere[i]]
        od ;
od ;
return itere ;
end ;
```

12. $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 1 & 5 & 4 \end{pmatrix}$ est de taille 5 mais d'ordre 6.

13.
```
pgcd :=proc(a,b)
local rest ;
rest :=reste(a,b) ;
if rest=0 then
        return b
else
        return pgcd(b,rest)
fi ;
end ;
```

14.
```
ppcm :=proc(a,b)
return a*b/pgcd(a,b) ; end ;
```

15.
```
ordreEfficace :=proc(t)
local perio,ordreE,i ;
perio :=periodes(t) ;
ordreE :=1 ;
for i from 1 to n do
        if reste(ordreE,perio[i])<>0 then
                ordreE :=ppcm(ordreE,perio[i])
        fi ;
od ;
return ordreE ;
end ;
```