a. Écrire une procédure qui donne la liste des diviseurs d'un entier en utilisant sa décomposition en produit de facteurs premiers. Quels sont les entiers parfaits, *i.e.*, égaux à la moitié de la somme de leurs diviseurs, inférieurs à 1000 ?

b. La fonction phi d'Euler est définie par

$$\varphi(n) = \mathrm{Card}(\mathbb{Z}/n\mathbb{Z})^\times$$

et elle vérifie

$$n = \sum_{d|n} \varphi(d).$$

Écrire une procédure qui la calcule.

c. Les polynômes cyclotomiques sont définis par

$$\Phi_n(X) = \prod_{k\in(\mathbb{Z}/n\mathbb{Z})^\times} \left(X - \exp\frac{2\pi ik}{n}\right)$$

et ils vérifient

$$X^n - 1 = \prod_{d|n} \Phi_d.$$

Écrire une procédure qui les calcule. On remarque que les coefficients de ces polynômes sont souvent -1, 0 ou -1 ; est-ce toujours le cas ?

a. Il est facile d'obtenir la liste de tous les diviseurs d'un entier en les essayant un par un.

```
restart;
diviseurs := proc(n::integer)
  local div,i;
  div:={};
  for i from 1 to n do
    if(irem(n,i)=0)
    then div:=div union {i};
    fi;
  od;
  RETURN(div);
end;
```

Mais c'est assez lent. On peut aussi utiliser la décomposition en produit de facteurs premiers : il faut alors procéder récursivement.

```
decomposition:=proc(m::integer)
```

```
  local dec,i,k,n;
  n:=m;
  dec:={};
  for i from 2 to floor(sqrt(n)) do
    if(irem(n,i)=0)
    then
      k:=0;
      while(irem(n,i)=0) do
        k:=k+1;
        n:=iquo(n,i);
      od;
      dec:= dec union {[i,k]};
    fi;
  od;
  if(n<>1)
  then dec:= dec union {[n,1]};
  fi;
  RETURN(dec);
end;
decomposition(4867416);
ifactor(4867416);

diviseurs:=proc(n::integer)
  _diviseurs(decomposition(n));
end;
_diviseurs:=proc(dec)
  local div,m,t,d,i;
  div:={};
  m:=nops(dec);
  if(m<>0)
  then
  t:=op(1,dec);
  for d in _diviseurs({op(2..m,dec)}) do
    for i from 0 to t[2] do
      div:= div union {d*t[1]^i};
    od;
  od;
  RETURN(div);
  else RETURN({1});
  fi;
end;
```

```
diviseurs(4867416);
```

On en déduit alors facilement les nombres parfaits inférieurs à 1000. Si on voulait aller plus loin, il faudrait prendre un « vrai » langage de programmation (qui nous donnerait les nombres parfait inférieurs à 100000 en deux fois moins de temps que Maple ceux inférieurs à 1000).

```
parfait:=proc(n::posint)
  evalb(convert(diviseurs(n),`+`)=2*n);
end;
parfait(6);
parfaits:=proc(n::posint)
  local i;
  for i from 2 to n do
    if(parfait(i))
    then print(i);
    fi;
  od;
end;
parfaits(100);
parfaits(1000);
```

b. La fonction φ d'Euler se calcule récursivement.

```
phi:=proc(n::posint)
  options remember;
  local f,d;
  f:=n;
  if(n=1)
  then RETURN(1);
  elif(isprime(n))
  then RETURN(n-1);
  else
    for d in diviseurs(n) do
      if(d<>n)
      then
        f:=f-phi(d);
      fi;
    od;
    RETURN(f);
  fi;
end;
phi(17);
phi(16);
```

c. Le code pour calculer les polynômes cyclotomiques est très semblable.

```
cyclo:=proc(n::posint)
  options remember;
  local c, d, i;
  c:=_X^n-1;
  if(n=1)
  then RETURN(_X-1);
  elif(isprime(n))
  then RETURN(sum(_X^i,i=0..n-1));
  else
    for d in diviseurs(n) do
      if(d<>n)
      then
        c:=normal(c/cyclo(d));
      fi;
    od;
    RETURN(c);
  fi;
end;
```

On peut maintenant regarder si leurs coefficients sont toujours 0, +1 ou -1.

```
toto:=proc(n::posint)
  local i;
  for i in {coeffs(cyclo(n))} do
    if(not member(i,{0,-1,+1}))
    then print(n,cyclo(n));
    fi;
  od;
end;
for k from 2 to 200 do
  toto(k);
od;
```

Ce n'est pas toujours le cas.