a. Implémenter l'algorithme de la division euclidienne. Généraliser cet algorithme pour résoudre l'équation $AU + BV = C$, où A, B et C sont des polynômes donnés et U et V les polynômes que l'on cherche.

b. Généraliser l'algorithme de la division aux polynômes à plusieures variables. L'algorithme nécessite d'ordonner les monômes : on prendra l'ordre lexicographique. Généraliser cet algorithme pour pouvoir diviser par plusieurs polynômes : on prend le premier et, quand on ne parvient plus à diviser, on passe au suivant ; l'algorithme s'arrête quand on ne peut plus diviser par aucun d'entre eux (c'est nécessaire, car si les idéaux de $k[x]$ sont tous principaux, *i.e.*, de la forme (P) pour un polynôme P, ceux de $k[x, y, z]$ ne peuvent pas (en général) être engendrés par un unique polynôme).

c. Essayer de diviser $x^2y + xy^2 + y^2$ par $xy - 1$ et $y^2 - 1$. On remarque que si on oublie le premier terme, on peut continuer la division. Implémenter ce prolongement de l'algorithme. Expérimenter sur quelques exemples : que se passe-t-il si on change l'ordre des diviseurs ? Les quotients sont-ils uniques ? Le reste est-il unique ?

d. Une base de Gröbner est une famille de polynômes $(f_1, \ldots, f_m)$ telle que le reste de la division d'un polynôme par cette famille soit unique. Pour transformer une famille en une base de Gröbner, on procède comme suit. On prend deux polynômes dans cette famille, f_i et f_j, on essaye de trouver une combinaison « linéaire » (à coefficients polynomiaux) qui fasse disparaître les termes dominants. On regarde ensuite le reste de la division de ce polynôme par notre famille. Si ce reste est non nul, on l'ajoute à la famille. On continue jusqu'à ce que tous ces restes soient nuls. Regarder ce qui se passe sur un exemple puis implémenter l'algorithme en général. Montrer, sur un exemple, que c'est une généralisation de l'algorithme du pivot de Gauss pour résoudre les systèmes d'équations linéaires.

e. On admet que pour obtenir une base de Gröbner minimale à partir d'une base de Gröbner G, il suffit de procéder ainsi : pour chaque élément de la base, on regarde si son terme dominant est dans l'idéal engendré par les termes dominants des autres éléments, si c'est le cas, on peut l'enlever. Écrire une procédure qui transforme une base de Gröbner en une base de Gröbner minimale.

f. On dit qu'une base de Gröbner est réduite si aucun monome d'aucun de ses éléments n'est dans l'idéal engendré par les termes dominants des autres éléments. Écrire une procédure qui transforme une base de Gröbner minimale en une base de Gröbner réduite. On peut montrer l'unicité de la base de Gröbner réduite d'un idéal (en demandant que les coefficients dominants des polynômes soient égaux à un).

g. À partir de maintenant, nous allons utiliser les commandes de la bibliothèque `grobner` de Maple, qui sont probablement plus rapides que les nôtres.
Le polynôme ... est-il dans l'idéal ...?

h. Montrer que les idéaux suivants sont égaux.

i. Montrer que l'algorithme de construction d'une base de Gröbner est une généralisation du pivot de Gauss.

j. Quels sont les points singuliers de la courbe d'équation ...?

k. Donner une équation de l'enveloppe de la famille de courbes ....

l. Trouver une équation implicite du tore.

m. Intersection de deux idéaux.

a. L'algorithme est bien connu. Pour que sa présentation ressemble à ce que nous allons faire tout-à-l'heure, nous commençons par définir une fonction qui nous dit si un polynôme en divise un autre et une fonction nous donnant le terme dominant d'un polynôme (et aussi une autre nous donnant le monome dominant).

```
divise:=proc(a::polynom,b::polynom)
  type(normal(b/a),polynom);
end;
lt:=proc(p::polynom)
  if(type(p,'+'))
  then
    sort(p);
    op(1,");
  else
    p;
  fi;
end;
lm:=proc(p::polynom,l::list)
  local q;
  q:=lt(p,l);
  if(type(op(1,q),numeric))
  then q/op(1,q);
  else q;
  fi;
end;
```

Voici maintenant l'algorithme.

```
division:=proc(f::polynom,g::polynom)
  local q,r;
  q:=0; r:=f;
  while( divise(lt(g),lt(r)) and (r<>0) ) do
    q:=q+lt(r)/lt(g);
    r:=expand( r-lt(r)/lt(g)*g );
  od;
  expand(q), r;
end;
```

On constate, sur un exemple, que cela semble marcher.

```
division(x^3+x-x^5, x+1);
"[1]*(x+1)+"[2];
expand(");
```

$$-x^5 + x^3 + x = (-x^4 + x^3 + 1) \cdot (x + 1) + 1$$

Pour résoudre l'équation $AU + BV = D$, on applique l'algorithme d'Euclide par récurrence. Si on a trouvé des polynômes U_n, V_n et D_n tels que $AU_n + BV_n = D_n$, on peut procéder par récurrence. Si la division euclidienne de D_n par D_{n+1} s'écrit $D_n = QD_{n+1} + R$, on obtient

$$\begin{aligned} R &= D_n - QD_{n+1} \\ &= AU_n + BV_n - Q(AU_{n+1} + BV_{n+1}) \\ &= A(U_n - QU_{n+1}) + B(V_n - QV_{n+1}), \end{aligned}$$

il suffit donc de poser

$$\begin{aligned} D_{n+2} &= R \\ U_{n+2} &= U_n - QU_{n+1} \\ V_{n+2} &= V_n - QV_{n+1} \end{aligned}$$

On démarre bien sûr la récurrence avec les identités $1 \cdot A + 0 \cdot B = A$ et $0 \cdot A + 1 \cdot B = B$. Le programme s'écrit alors comme suit.

```
aubv:=proc(A::polynom, B::polynom)
  local u,uu,uuu,v,vv,vvv,d,dd,ddd,q,r;
  u:=1;  v:=0;  d:=A;
  uu:=0; vv:=1; dd:=B;
  while(dd<>0) do
    q:=0; r:=d;
    while( divise(lt(dd),lt(r)) and (r<>0) ) do
```

```
      q:=q+lt(r)/lt(dd);
      r:=expand( r-lt(r)/lt(dd)*dd );
    od;
    ddd:=d-q*dd; uuu:=u-q*uu; vvv:=v-q*vv;
    d:=expand(dd); dd:=expand(ddd);
    u:=expand(uu); uu:=expand(uuu);
    v:=expand(vv); vv:=expand(vvv);
  od;
  u, v, d, u*A+v*B=d;
end;
```

On vérifie sur un exemple qu'il semble correct.

```
aubv(x^2-1, x^2+2*x+1);
expand("[4]);
```

$$(x^2 - 1) - (x^2 + 2x + 1) = -2x - 2$$

Cela nous donne le pgcd des deux polynômes avec une solution de l'équation. Je laisse au lecteur le soin de les trouver toutes.

b. Pour les polynômes à pluisieures variables, le seul changement est la fonction `lt` donnant le terme dominant. Elle attend un argument supplémentaire, l'ordre les variables. Dans les exemple qui suivent, nous prendrons $x \geqslant y \geqslant z$.

```
lt:=proc(p::polynom,l::list)
  if(type(p,'+'))
  then
    sort(p,l,plex);
    op(1,");
  else
    p;
  fi;
end;
```

La procédure effectuant la division est la même que ci-dessus.

```
division:=proc(f::polynom,g::polynom,l::list)
  local q,r;
  q:=0; r:=f;
  while( divise(lt(g,l),lt(r,l)) and (r<>0) ) do
    q:=q+lt(r,l)/lt(g,l);
    r:=expand( r-lt(r,l)/lt(g,l)*g );
  od;
  expand(q), r;
end;
```

Vérifions sur un exemple que cela semble marcher.

```
division(x^2+y^2+z^2, x+1, [x,y,z]);
(x+1)*"[1]+"[2]; expand(");
```

$$x^2 + y^2 + z^2 = (x-1)(x+1) + (y^2 + z^2 + 1)$$

Ajoutons maintenant la possibilité de diviser par une famille de polynômes.

```
division:=proc(f::polynom,g::list,l::list)
  local r,q,i,j,n, jaidivise;
  n:=nops(g); r:=f;
  for i from 1 to n do q[i]:=0 od;
  jaidivise:=true;
  while(jaidivise) do
    jaidivise:=false;
    for i from 1 to n while(not jaidivise) do
      if divise(lt(g[i],l), lt(r,l)) then
        q[i]:=q[i]+lt(r,l)/lt(g[i],l);
        r:=expand( r-lt(r,l)/lt(g[i],l)*g[i] );
        jaidivise:=true;
      fi;
    od;
  od;
  seq(q[j],j=1..n), r;
end;
```

Vérifions sur un exemple que cela semble marcher.

```
f:=x*y^2+1; g1:=x*y+1; g2:=y+1;
division(f,[g1,g2],[x,y]);
"[1]*g1+"[2]*g2+"[3];
expand(");
```

$$xy^2 + 1 = y(xy+1) - (y+1) + 2$$

c. Dans l'exemple de l'énoncé,

```
f:=x^2*y+x*y^2+y^2; g1:=x*y-1; g2:=y^2-1;
division(f,[g1,g2],[x,y]);
```

$$x^2y + xy^2 + y^2 = (x+y)(xy-1) + 0 \cdot (y^2-1) + x + y^2 + y,$$

on a envie de mettre de côté le premier terme du reste, ici x, afin de poursuivre la division. Programmons cette amélioration de l'algorithme.

```
division:=proc(f::polynom,g::list,l::list)
local a,r,p,i,did_it,s,j;
  p:=f;
```

```
    s:=nops(g);
    for i from 1 to s do a[i]:=0; od;
    r:=0;
    while(p<>0) do
      i:=1;
      did_it:=false;
      while((i<=s) and not(did_it)) do
        if divise( lt(g[i],l), lt(p,l) )
        then
          a[i]:=a[i]+lt(p,l)/lt(g[i],l);
          p:=p-lt(p,l)/lt(g[i],l)*g[i];
          p:=expand(p);
          did_it:=true;
        else
          i:=i+1;
        fi;
      od;
      if(not did_it)
      then
        r:=r+lt(p,l);
        p:=expand(p-lt(p,l));
      fi;
    od;
    seq(a[j],j=1..s),r;
  end;
```

Reprenons le même exemple.

```
  f:=x^2*y+x*y^2+y^2; g1:=x*y-1; g2:=y^2-1;
  division(f,[g1,g2],[x,y]);
  f="[1]*g1+"[2]*g2+"[3];
  expand(");
```

$$x^2y + xy^2 + y^2 = (x + y)(xy - 1) + (y^2 - 1) + x + y + 1$$

```
  division(f,[g2,g1],[x,y]);
  f="[1]*g2+"[2]*g1+"[3];
  expand(");
```

$$x^2y + xy^2 + y^2 = (x + 1)(y^2 - 1) + x(xy - 1) + 2x + 1$$

On constate que le reste et les quotients ne sont pas déterminés de manière unique : si on change l'ordre des diviseurs, on peut avoir un reste différent.

L'exemple suivant nous montre que cela ne fournit même pas un critère d'appartenance à un idéal : dans le premier cas, le reste est nul (on sait donc que f est dans l'idéal $\langle g_1, g_2\rangle$), dans le second, il ne l'est pas (on ne sait rien).

```
f:=x*y^2-x;
g1:=y^2-1;
g2:=x*y+1;
division(f,[g1,g2],[x,y]);
```

$$xy^2 - x = x(y^2-1) + 0\cdot(xy+1) + 0$$

```
division(f,[g2,g1],[x,y]);
```

$$xy^2 - x = y(xy+1) + 0\cdot(y^2-1) + -x - y$$

d. On élimine les termes de plus haut degré comme suit (on parle de « polynôme S »).

```
s:=proc(p::polynom,q::polynom,l::list)
  (p/lt(p,l) - q/lt(q,l))*lcm(lm(p,l),lm(q,l));
  expand(");
end;
```

Commençons par construire une base de Gröbner à la main, sur un exemple. Partons des deux polynômes suivants.

```
f1:=x^3-2*x*y;
f2:=x^2*y-2*y^2+x;
```

Calculons le polynôme S correspondant et regardons le reste de la division : comme il est non nul, on l'ajoute à la famille.

```
reste:=proc(f::polynom,l::list)
  division(f,l,[x,y]); "[nops(["])];
end;
s(f1,f2,[x,y]);
```

$$\left(\frac{x^3-2xy}{x^3} - \frac{x^2y-2y^2+x}{x^2y}\right)x^3y = -x^2$$

```
reste(",[f1,f2]);
f3:=";
```

On continue, on rajoute les polynômes un à un, à chaque fois que le reste est non nul.

```
s(f1,f3,[x,y]); reste(",[f1,f2,f3]);
f4:=";
s(f2,f3,[x,y]); reste(",[f1,f2,f3,f4]);
f5:=";
```

```
s(f1,f5,[x,y]); reste(",[f1,f2,f3,f4,f5]);
f6:=";
```
On peut maintenant vérifier que l'on ne peut plus rien rajouter. D'après le résultat admis dans l'énoncé, on sait que cela suffit pour avoir une base de Gröbner.
```
s(f1,f2,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f1,f3,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f1,f4,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f1,f5,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f1,f6,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f2,f3,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f2,f4,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f2,f5,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f2,f6,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f3,f4,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f3,f5,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f3,f6,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f4,f5,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f4,f6,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
s(f5,f6,[x,y]); reste(",[f1,f2,f3,f4,f5,f6]);
```
La base de Grobner est donc

$$x^3-2xy, x^2y+x-2y^2, -x^2, -2xy, x-2y^2, 4y^3.$$

Comme c'est particulièrement répétitif, nous allons mettre cela dans une boucle. L'algorithme général d'obtention d'une base de Gröbner est alors le suivant.
```
reste:=proc(f::polynom,l::list,v::list)
  division(f,l,v); "[nops(["])];
end;
grob:=proc(F::list,l::list)
  local G,GG,fini,p,q,S;
  G:=copy(F);
  fini:=false;
  while(not fini) do
    GG:=copy(G);
    fini:=true;
    for p in GG do
      for q in GG do
        if( p<>q ) then
          S:=s(p,q,l);
```

```
        S:=reste(S,G,l);
        if( S<>0 ) then
          G:=[op(G), S];
          fini:=false;
        fi;
      fi;
    od;
  od;
od;
G;
end;
```

On peut comparer le résultat de cette procédure avec nos calculs manuels précédents.

```
grob([f1,f2],[x,y]);
[f1,f2,f3,f4,f5,f6];
```

Ainsi, nous pouvons vérifier que le polynôme x^10 est bien dans l'idéal engendré par f_1 et f_2, bien que le reste de sa division par f_1 et f_2 soit non nul.

```
reste(x^10,[f1,f2]);
```

$$-16xy^3 + 32y^5$$

```
reste(x^10,[f2,f1]);
```

$$-64xy^3 - 40x + 32y^5 + 80y^2$$

```
reste(x^10,[f1,f2,f3,f4,f5,f6]);
```

$$0$$

e. Commençons par écrire une procédure qui nous dit si on doit enlever un polynôme à une base de Gröbner, dans l'algorithme d'obtention d'une base minimale. Pour savoir si le terme dominant est dans l'idéal engendré par les termes dominants, il suffit de remarquer qu'un ensemble de monômes forment une base de Gröbner (car les S-polynômes sont tous nuls) : c'est le reste de la division du terme dominant par les autres termes dominants qui nous dit s'il est dans l'idéal ou non.

```
enleve:=proc(g::set,p::polynom,l::list)
  local m,i;
  m:={};
  for i in g do
    if(i<>p) then m:=m union{lt(i,l)} fi;
```

```
    od;
    reste(lt(p,l),[op(m)],l)=0;
  end;
```
On peut vérifier que, dans notre exemple, on peut enlever f_2 ou f_4 (et ce ne sont pas les seuls).
```
  enleve({f1,f3,f4,f5,f6},f2,[x,y]);
  enleve({f1,f2,f3,f5,f6},f4,[x,y]);
```
Il suffit maintenant de mettre cela dans une boucle.
```
  minimal:=proc(g::set,l::list)
    local p,m;
    m:=g;
    for p in g do
      if( enleve(m,p,l) )
      then
        m:=m minus {p};
      fi;
    od;
    m;
  end;
  minimal({f1,f2,f3,f4,f5,f6},[x,y]);
```
Sur notre exemple, on constate que la base de Gröbner minimale n'est constituée que de deux éléments, $x - 2y^2$ et $4y^3$.

f. Expliquer la procédure suivante.
```
  reduit:=proc(g::set,l::list)
    local r,p,q;
    r:=minimal(g,l);
    for p in r do
      q:=reste(p,[op(r minus {p})],l);
      r:=(r minus {p}) union {q};
    od;
    r;
  end;
```
On constate que dans l'exemple précédent, la base de Gröbner minimale obtenue était déjà réduite. Pour vérifier que notre procédure fonctionne, on peut la modifier, pour en obtenir une minimale non réduite.
```
  reduit({f1,f2,f3,f4,f5,f6},[x,y]);
  reduit({x-2*y^2 + y^4, 4*y^3},[x,y]);
```
$$(x - 2y^2 + y^4, 4y^3) = (x - 2y^2, y^3)$$