I Algorithme d'Euclide classique sur les polynÙmes

```
> euclide1:=proc(A,B)
  local U,V,A1,B1,T,R,Q;
  U:=1;A1:=A;B1:=B;V:=0;
  while B1 <> 0 do

      R:=rem(A1,B1,X);Q:=quo(A1,B1,X);
      print(R);
      T:=U-Q*V;U:=V;A1:=B1;V:=T;B1:=R;
  od;
  if divide((A1-A*U),B,'q') then
      V:=q;
  fi;
  [U,V,expand(A1)];
  end;
```

```
> f:=824*X^5-65*X^4-814*X^3-741*X^2-979*X-764;
```

```
> g:=216*X^4+663*X^3+880*X^2-916*X+617;
```

```
> euclide1(f,g);
```

II Algorithme d'Euclide "unitaire" sur les polynÙmes

```
> euclide2:=proc(A,B)
  local U,V,A1,B1,T,R,Q;
  U:=1;A1:=A;B1:=B;V:=0;
  while B1 <> 0 do
      R:=rem(A1,B1,X);
      if R<>0 then
            R:=R/lcoeff(R);
            print(R);
      fi;
      Q:=quo(A1,B1,X);
      T:=U-Q*V;U:=V;A1:=B1;V:=T;B1:=R;
  od;
  if divide((A1-A*U),B,'q') then
      V:=q;
  fi;
  expand(A1);
  end;
```

```
> euclide2(f,g);
```

III Algorithme d'Euclide primitif

renvoit le coefficient dominant de f si f<>0 et 1 sinon.

```
> lu:=proc(f)
  local l;
  l:=1;
  if f<>0 then
     l:=lcoeff(f,X);
  fi;
  l;
  end;
```

Algorithme d'Euclide avec des opÈrations exclusivement dans Z[X]
(on utilise des pseudo divisions dan Z[X]).

```
> euclideprimitif:=proc(f,g)
  local f1,g1,n1,n2,i,t;
  f1:=f;
  g1:=g;
  n1:=degree(f,X);
  n2:=degree(g,X);
  while g1 <> 0 do
      t:=rem(lcoeff(g1,X)^(1+n1-n2)*f1,g1,X);
      n1:=n2;
      f1:=g1;
      print(f1);
      g1:=t;
      n2:=degree(g1,X);
  od;
  f1/lu(f1);
  end;
```

Application ‡ l'exemple du texte.

```
> euclideprimitif(f,g);
```

Un autre exemple...

```
euclideprimitif((X-1)^3*x^9*13,(X-1)^2*13);
```

IV Algorithme d'Euclide modulaire "version grand nombre premier"

Le contenu d'un polynome (i.e. le pgcd des ses coefficients)
avec pour convention cont(0) = 1.

```
> cont:=proc(f)
  local c,n,i;
  c:=1;
  if f <>0 then
     n:=degree(f,X);
     c:=coeff(f,X,0);
     for i from 1 to n while c<> 1 do
     c:=igcd(c,coeff(f,X,i));
     od;
  fi;
  c;
  end;
```

```
> cont(2*X^7+2*X);
```

Partie primitive d'un polynome.
Renvoit le polynome de contenu 1 du meme degrÈ que f qui divise f.

```
> primitivepart:=proc(f)
  f/cont(f);
  end;
```

```
> primitivepart(f);
```

Programmation de la norme infini sur les coefficients d'un polynome de C[X].
On peut utiliser et on utilisera la fonction maple norm(f,infinity).

```
> normInfini:=proc(P)
  local n,N,i,t;
  n:=-1;
  N:=0;
  if P<>0 then
      n:=degree(P);
  fi;
  for i from 0 to n do
      t:=abs(coeff(P,X,i));
      if evalf(t)>N then
          N:=t;
      fi;
  od;
  N;
  end;
```

```
> f2:=(1+I)*X+1;
```

```
> normInfini(f2);
```

Donne un nombre premier choisi au hasard entre les entiers 2*B et 4*B avec B donnÈ en argument.

```
> premier:=proc(B)
  local p,ra,k;
  ra:=rand(2*B..4*B);
  p:=4*B+1;
  while p>(4*B) do
  k:=ra();
```

```
   p:=nextprime(k);
   od;
   p;
   end;
```

```
> premier(1000);
```

Donne la forme normale d'un polynome dans Z/pZ [X],
(c'est ‡ dire si f<>0, le polynome unitaire qui divise f* et qui est de mÍme degrÈ que f* avec f = f* mod p).

```
> formeNormal:=proc(f,p)
   local n,k,f1,a;
   f1:=0;
   if f<>0 then
      n:=degree(f mod p,X);
      igcdex(lcoeff(f mod p) mod p,p,'u','v');
      a:=u mod p;
      for k from 0 to n do
          f1:=f1+X^k*a*coeff(f,X,k) mod p;
      od;
   fi;
   f1;
   end;
```

```
> formeNormal(g,3);
```

Algorithme d'Euclide pour calculer le polynÙme unitaire v de Z [X] tel que normInfiny(v)<p/2 et tel que v mod p = pgcd (f mod p, g mod p);

```
> euclideModP:=proc(f,g,p)
   local k,r,q,s,t,l,i,a;
   k[0]:=lu(f);
   r[0]:=formeNormal(f mod p,p);
   igcdex(k[0],p,'u','v');
   s[0]:=u;
   t[0]:=0;
   k[1]:=lu(g mod p);
   r[1]:=formeNormal(g,p);
   igcdex(k[1],p,'u','v');
   t[1]:=u;
   s[1]:=0;
   i:=1;
   while r[i]<>0 do
      q[i]:=quo(r[i-1],r[i],X);
      k[i+1]:=lu((expand(r[i-1]-q[i]*r[i])) mod p);
      igcdex(k[i+1],p,'u','v');
      a:=u;
      r[i+1]:=expand((r[i-1]-q[i]*r[i])*a) mod p;
      s[i+1]:=expand((s[i-1]-q[i]*s[i])*a) mod p;
      t[i+1]:=expand((t[i-1]-q[i]*t[i])*a) mod p;
      i:=i+1;
   od;
   l:=i-1;
   s[l]:=mods(s[l],p);
   t[l]:=mods(t[l],p);
   r[l]:=mods(r[l],p);
```

```
   [s[l],t[l],r[l]];
   end;
```

```
> euclideModP(X-3,X+3,3);
```

```
> euclideModP(f,g,3);
```

Algorithme d'Euclide modulaire "version grand nombre premier".
EntrÈe: f,g deux polynÙmes primitifs de Z[X] avec n:=deg f >= deg g >= 1 et normInfini(f), normInfini(b) <= A.
Sortie: h:=pgcd(f,g) dans Z[X]

```
> euclide3:=proc(f,g)
   local A,n,h,p,b,B,f1,g1,v,w;
   n:=degree(f,X);
   b:=igcd(lcoeff(f,X),lcoeff(g,X));
   A:=max(ceil(evalf(normInfini(f))),ceil(evalf(normInfini(g))))
   ;
   B:=ceil(evalf((n+1)^(1/2)*2^n*A*b));
   w:=B+1;
   f1:=1;
   g1:=1;
   while norm(f1,1)*norm(w,1)>B or norm(g1,1)*norm(w,1)>B do
       p:=premier(B);
       v:=euclideModP(f,g,p)[3];
       w:=mods(b*v,p);
       f1:=mods(expand(Quo(b*f,w,X) mod p),p);
       g1:=mods(expand(Quo(b*g,w,X)mod p),p);
   od;
   primitivepart(w);
   end;
```

```
> euclide3(f,g);
```

```
> A:=expand(randpoly([X],degree = 30)*(X^2+1));
```

```
> B:=expand(randpoly([X],degree = 30)*(X^2+1));
```

```
> euclide3(A,B);
```

```
>
```

```
>
```