DS07032018

```
def rev(kx,i,F):
    if i>=F.degree():
        return kx(x^i*F(x^(-1)))
    else:
        return "Erreur"
```

```
def Inverse1(P,Q):
    d,u,v=xgcd(P,Q)
    if d<>1:
        return "Erreur"
    else:
        return u
```

```
kx.<x>=PolynomialRing(GF(17))
```

```
F=sum([i*x^i for i in range(1,6)])
```

```
F
```

```
    5*x^5 + 4*x^4 + 3*x^3 + 2*x^2 + x
```

```
G=x^2+2*x+3
```

```
G
```

```
    x^2 + 2*x + 3
```

```
G0=rev(kx,2,G)
```

```
G0
```

```
    3*x^2 + 2*x + 1
```

```
G1=Inverse1(G0,x^4)
```

```
G1
```

```
    4*x^3 + x^2 + 15*x + 1
```

```
Q1=(rev(kx,5,F)*G1)%x^4
```

```
Q1
```

```
    3*x^3 + 11*x + 5
```

```
Q=rev(kx,3,Q1)
```

```
Q
```

```
    5*x^3 + 11*x^2 + 3
```

```
F-G*Q
```

```
    12*x + 8
```

```
def Inverse2(F,n):
    A=1
    k=1
    while k<n:
        k=2*k
        F_red=F % x^k
        A2=(A^2) % x^k
```

```
        FA2=(F_red*A2) % x^k
        A=2*A-FA2
    return A % x^n
```

```
Inverse2(G0,5)
```

```
    6*x^4 + 4*x^3 + x^2 + 15*x + 1
```

```
_*G0
```

```
    x^6 + 7*x^5 + 1
```

```
def Division(kx,F,G):
    m,n=F.degree(),G.degree()
    if m<n:
        return (0,F)
    else:
        G1=rev(kx,n,G)
        G1=Inverse2(G1/G1(0),m-n+1)/G1(0)
        Q1=(rev(kx,m,F)*G1)%x^(m-n+1)
        Q=rev(kx,m-n,Q1)
        return Q,F-G*Q
```

```
Div=Division(kx,F,G);Div
```

```
    (5*x^3 + 11*x^2 + 3, 12*x + 8)
```

```
F;G;G*Div[0]+Div[1]
```

```
    5*x^5 + 4*x^4 + 3*x^3 + 2*x^2 + x
    x^2 + 2*x + 3
    5*x^5 + 4*x^4 + 3*x^3 + 2*x^2 + x
```

```
F.quo_rem(G)
```

```
    (5*x^3 + 11*x^2 + 3, 12*x + 8)
```