Maple generic code with Domains package

```
GCD := proc(A,B,P,G,phi,NextPrime,SymRem)

  if A = P[0] then RETURN( P[Normal](B) ) fi;
  if B = P[0] then RETURN( P[Normal](A) ) fi;

  E := P[CoefficientRing]; # Euclidean domain
  R := G[CoefficientRing]; # Residue ring
  X := P[ExponentVector]; # Ordered Abelian monoid

  a := P[Primpart](A,’ca’);
  b := P[Primpart](B,’cb’);
  gc := E[Gcd](ca,cb); # GCD of contents

  da := P[Degree](a);
  db := P[Degree](b);
  degbound := X[Min](da,db);

  la := P[Lcoeff](a);
  lb := P[Lcoeff](b);
  gamma := E[Gcd](la,lb);

  hbar := P[0]; modulus := E[1];
  for k do

    R[Modulus] := NextPrime();
    while phi(gamma) = E[0] do
      R[Modulus] := NextPrime()
    od;
    m := R[Modulus];

    abar := P[Map](phi,a);
    bbar := P[Map](phi,b);
    userinfo(2,Gcd,"Image GCD computation");
    gbar := G[Gcd](abar,bbar);
    d := G[Degree](gbar); # vector degree
    if d = X[0] then RETURN( P[Constant](gc) ) fi;

    # Leading coefficient correction
    # gbar is assumed to be monic
    gbar := G[`.`]( phi(gamma), gbar );

    if hbar = P[0] or X[`<`](d,degbound) then
       # All previous homomorphism's were unlucky
       degbound := d;
       hbar := P[0];
       modulus := E[1];

    elif X[`>`](d,degbound) then
       # This homomorphism is unlucky
       next;
    fi;

    userinfo(2,Gcd,"Chinese remaindering");
    s := R[Inv](phi(modulus));
    v1 := G[`.`](s,G[`-`](gbar,P[Map](phi,hbar)));
    h := P[`+`](hbar,P[`.`](modulus,v1));

    modulus := E[`*`](m,modulus);
    h := P[Map](proc(x) SymRem(x,modulus) end, h);

    if h = hbar then
      userinfo(2,Gcd, "Beginning termination check");
      g := P[Primpart](h);
      if P[Divides](g,a) and P[Divides](g,b) then
# Unit normalization is not multiplicative in Z[i]
        RETURN( P[Normal](P[`.`](gc,g)) );
      fi;
    fi;
    hbar := h;
  od;

end:
```