# Question 5d:

```
> restart;
> interface(prettyprint=0);
0
> (* Comp_rem procedure computes the remainder and quotient of two
  gaussian integers *)
> comp_rem := proc(a::complex, b::complex)

      (* Initializing local variables *)

      local comp_eval, comp_q, comp_r, A, B;

      A  :=  a;
      B  :=  b;
      comp_eval := evalf(A/B);

      (* Formula based on part(c) *)

     comp_q := round(Re(comp_eval)) + I*round(Im(comp_eval));
     comp_r := A - B*comp_q;

      return comp_q, comp_r;
  end proc;
Typesetting:-mprintslash([(comp_rem := proc (a::complex, b::complex)
local
comp_eval, comp_q, comp_r, A, B; A := a; B := b; comp_eval := evalf
(A/B);
comp_q := round(Re(comp_eval))+I*round(Im(comp_eval)); comp_r := A-B*
comp_q;
return comp_q, comp_r; end proc)],[proc (a::complex, b::complex)
local
comp_eval, comp_q, comp_r, A, B; A := a; B := b; comp_eval := evalf
(A/B);
comp_q := round(Re(comp_eval))+I*round(Im(comp_eval)); comp_r := A-B*
comp_q;
return comp_q, comp_r; end proc])

> (* Comp_gcd computes the gcd of two gaussian integers *)
> comp_gcd := proc(a::complex, b::complex)

      (* Initializing local variables *)

      local A, B, comp_q, comp_r, unit_list;

      (* Units of gaussian integers *)
      unit_list := [1, -1, I, -I];
      A := a;
      B := b;

      (* Recursive loop to compute the gcd until B = 0 *)

      while B <> 0 do
      (comp_q, comp_r) := comp_rem(A,B);
      A := B;
      B := comp_r;
      od;

      (* Multiplication by a unit to get the positive gcd *)

     if Re(A) < 0 then
     A := unit_list[2] * A;
      end if;

      return A;
  end proc;
```

```
Typesetting:-mprintslash([(comp_gcd := proc (a::complex, b::complex)
local A, B
, comp_q, comp_r, unit_list; unit_list := [1, -1, I, -I]; A := a; B
:= b;
while B <> 0 do comp_q, comp_r := comp_rem(A,B); A := B; B := comp_r;
end do;
if Re(A) < 0 then A := unit_list[2]*A; end if; return A; end proc)],
[proc (a::
complex, b::complex) local A, B, comp_q, comp_r, unit_list; unit_list
:= [1, -1
, I, -I]; A := a; B := b;  while B <> 0 do comp_q, comp_r := comp_rem
(A,B); A
:= B; B := comp_r; end do; if Re(A) < 0 then A := unit_list[2]*A; end
if;
return A; end proc])
```

```
> A := 63+10*I;
  B := 7+43*I;
  test_rec_one := comp_gcd(A, B);
  comp_rem(A,B);
```

```
Typesetting:-mprintslash([(A := 63+10*I)],[63+10*I])
Typesetting:-mprintslash([(B := 7+43*I)],[7+43*I])
Typesetting:-mprintslash([(test_rec_one := 2+3*I)],[2+3*I])
-I, 20+17*I
```

```
>
  C := 330;
  E := -260;
  test_rec_two := comp_gcd(C, E);
Typesetting:-mprintslash([(C := 330)],[330])
Typesetting:-mprintslash([(E := -260)],[-260])
Typesetting:-mprintslash([(test_rec_two := 10)],[10])
```

## (Aside) CPU Usage:

```
> CodeTools:-Usage(comp_gcd(A, B) ) ;
memory used=38.27KiB, alloc change=0 bytes, cpu time=2.00ms, real
time=2.00ms, gc time=0ns
2+3*I
```

## (Aside) Confirming answer with Gauss Package:

```
> check_one := GaussInt:-GIgcd(A, B);
  check_two := GaussInt:-GIgcd(C, E);
Typesetting:-mprintslash([(check_one := 2+3*I)],[2+3*I])
Typesetting:-mprintslash([(check_two := 10)],[10])
```

```
> CodeTools:-Usage(GaussInt:-GIgcd(A, B));
memory used=9.19KiB, alloc change=0 bytes, cpu time=0ns, real time=
0ns, gc time=0ns
2+3*I
```