Question 4:

```
> restart;
> interface(prettyprint=0);
0

> (* Swapping 2 numbers procedure. The function takes in two
  arguments a and b of type float and swaps them. *)
> swap_num := proc(a::posint, b::posint)

      (* Initializing local variables *)

     local temp_var, A, B;
      A := a;
      B := b;
     temp_var := A;
      A := B;
     B := temp_var;

     return A,B;
  end proc;
Typesetting:-mprintslash([(swap_num := proc (a::posint, b::posint)
local
temp_var, A, B; A := a; B := b; temp_var := A; A := B; B := temp_var;
return A,
B; end proc)],[proc (a::posint, b::posint) local temp_var, A, B; A :=
a; B := b
; temp_var := A; A := B; B := temp_var; return A, B; end proc])

> (* Binary GCD Procedure. The function takes in two arguments a
  and b that are positive integers and
    outputs the gcd of a and b.*)
> bin_gcd := proc(a::posint, b::posint)

  (* Initializing local variables *)

  local counter_var, A, B;
  counter_var := 0;
  A := a;
  B := b;

  (* Terminating condition if values of A or B = 0 *)

  if A = 0 then
  return B;
  elif B = 0 then
  return A;
  end if;
```

```
  (* Condition 1: If both A and B have a factor of 2 in common, we
  factor out the 2 and increment count
  by +1 *)

  while irem(A,2) = 0 and irem(B,2) = 0 do
  A := (A/2);
  B := (B/2);
  counter_var := counter_var+1;
  printf("Value of A and B: %d, %d\n\n", A,B);
  od;

  (* For the rest of the conditions *)

  while A <> 0 do

  (* Factoring out 2 for both A and B *)

  while irem(A,2) = 0 do
  A := (A/2);
  printf("Value of A and B: %d, %d\n\n", A,B);
  od;

  while irem(B,2) = 0 do
  B := (B/2);
  printf("Value of A and B: %d, %d\n\n", A,B);
  od;

  (* To make sure A is always > B *)

  if A < B then
  A,B := swap_num(A,B);
  printf("Value of A and B: %d, %d\n\n", A,B);
  end if;

  A := A-B;
  printf("Value of A and B: %d, %d\n\n", A,B);
  od;

  (* Function returns the value of B times the number of times we
  factored out 2 *)

  return B * 2^counter_var, counter_var;
  end proc;
Typesetting:-mprintslash([(bin_gcd := proc (a::posint, b::posint)
local
counter_var, A, B; counter_var := 0; A := a; B := b; if A = 0 then
return B;
elif B = 0 then return A; end if;  while irem(A,2) = 0 and irem(B,2)
```

```
= 0 do A
:= 1/2*A; B := 1/2*B; counter_var := counter_var+1; printf(
"Value of A and B: %d, %d\n\n",A,B); end do;  while A <> 0 do  while
irem(A,2)
= 0 do A := 1/2*A; printf("Value of A and B: %d, %d\n\n",A,B); end
do;  while
irem(B,2) = 0 do B := 1/2*B; printf("Value of A and B: %d, %d\n\n",A,
B); end do
; if A < B then A, B := swap_num(A,B); printf("Value of A and B: %d,
%d\n\n",A,
B); end if; A := A-B; printf("Value of A and B: %d, %d\n\n",A,B); end
do;
return B*2^counter_var, counter_var; end proc)],[proc (a::posint,
b::posint)
local counter_var, A, B; counter_var := 0; A := a; B := b; if A = 0
then return
B; elif B = 0 then return A; end if;  while irem(A,2) = 0 and irem(B,
2) = 0 do
A := 1/2*A; B := 1/2*B; counter_var := counter_var+1; printf(
"Value of A and B: %d, %d\n\n",A,B); end do;  while A <> 0 do  while
irem(A,2)
= 0 do A := 1/2*A; printf("Value of A and B: %d, %d\n\n",A,B); end
do;  while
irem(B,2) = 0 do B := 1/2*B; printf("Value of A and B: %d, %d\n\n",A,
B); end do
; if A < B then A, B := swap_num(A,B); printf("Value of A and B: %d,
%d\n\n",A,
B); end if; A := A-B; printf("Value of A and B: %d, %d\n\n",A,B); end
do;
return B*2^counter_var, counter_var; end proc])
```

```
> A := 16*3*101;
  B := 8*3*203;

  test_res_one, total_counter := bin_gcd(A, B);
  printf("\nThe GCD is: %d and the factor count is: %d.",
  test_res_one, total_counter);
```

```
Typesetting:-mprintslash([(A := 4848)],[4848])
Typesetting:-mprintslash([(B := 4872)],[4872])
Value of A and B: 2424, 2436

Value of A and B: 1212, 1218

Value of A and B: 606, 609

Value of A and B: 303, 609

Value of A and B: 609, 303
```

```
Value of A and B: 306, 303

Value of A and B: 153, 303

Value of A and B: 303, 153

Value of A and B: 150, 153

Value of A and B: 75, 153

Value of A and B: 153, 75

Value of A and B: 78, 75

Value of A and B: 39, 75

Value of A and B: 75, 39

Value of A and B: 36, 39

Value of A and B: 18, 39

Value of A and B: 9, 39

Value of A and B: 39, 9

Value of A and B: 30, 9

Value of A and B: 15, 9

Value of A and B: 6, 9

Value of A and B: 3, 9

Value of A and B: 9, 3

Value of A and B: 6, 3

Value of A and B: 3, 3

Value of A and B: 0, 3

Typesetting:-mprintslash([(test_res_one, total_counter := 24, 3)],
[24, 3])

The GCD is: 24 and the factor count is: 3.
```

(Aside) : CPU Usage Statistics

```
> CodeTools:-Usage(bin_gcd(A, B)):
Value of A and B: 2424, 2436

Value of A and B: 1212, 1218

Value of A and B: 606, 609

Value of A and B: 303, 609

Value of A and B: 609, 303

Value of A and B: 306, 303

Value of A and B: 153, 303

Value of A and B: 303, 153

Value of A and B: 150, 153

Value of A and B: 75, 153

Value of A and B: 153, 75

Value of A and B: 78, 75

Value of A and B: 39, 75

Value of A and B: 75, 39

Value of A and B: 36, 39

Value of A and B: 18, 39

Value of A and B: 9, 39

Value of A and B: 39, 9

Value of A and B: 30, 9

Value of A and B: 15, 9

Value of A and B: 6, 9

Value of A and B: 3, 9

Value of A and B: 9, 3

Value of A and B: 6, 3
```

```
Value of A and B: 3, 3

Value of A and B: 0, 3

memory used=76.70KiB, alloc change=0 bytes, cpu time=1000.00us, real
time=2.00ms, gc time=0ns
```

(Aside) Testing with random integers (range = 1 to 10^11)

```
> roll := rand(1..100000000000);
Typesetting:-mprintslash([(roll := proc () proc () option builtin =
RandNumberInterface; end proc(6,100000000000,37)+1; end proc)],[proc
() proc ()
option builtin = RandNumberInterface; end proc(6,100000000000,37)+1;
end proc])
```

```
> gcd_test := proc()
    local num_one, num_two, i, res, dev_algo;
    local start_time_res, stop_time_res, start_time_dev,
  stop_time_dev;
    local size_list, bin_time_list, gcd_time_list, n;

     n := 10;
    size_list := Array(1..n);
    bin_time_list := Array(1..n);
    gcd_time_list := Array(1..n);

    for i from 1 to n do
       num_one := roll();
       num_two := roll();

      printf("***** Testing for numbers: %d and %d ******\n\n",
  num_one, num_two);

       start_time_res := time();
      res := bin_gcd(num_one, num_two):
       stop_time_res := time() - start_time_res;

       start_time_dev := time();
      dev_algo := gcd(num_one, num_two):
       stop_time_dev := time() - start_time_dev;

      size_list[i] := max(abs(num_one), abs(num_two));
       bin_time_list[i] := stop_time_res;
       gcd_time_list[i] := stop_time_dev;
```

```
        printf("Computed GCD is: %d.\n\n", res):
      od;
  end proc;
Typesetting:-mprintslash([(gcd_test := proc () local num_one,
num_two, i, res,
dev_algo, start_time_res, stop_time_res, start_time_dev,
stop_time_dev,
size_list, bin_time_list, gcd_time_list, n; n := 10; size_list :=
Array(1 .. n)
; bin_time_list := Array(1 .. n); gcd_time_list := Array(1 .. n); for
i to n do
num_one := roll(); num_two := roll(); printf(
"***** Testing for numbers: %d and %d ******\n\n",num_one,num_two);
start_time_res := time(); res := bin_gcd(num_one,num_two);
stop_time_res :=
time()-start_time_res; start_time_dev := time(); dev_algo := gcd
(num_one,
num_two); stop_time_dev := time()-start_time_dev; size_list[i] := max
(abs(
num_one),abs(num_two)); bin_time_list[i] := stop_time_res;
gcd_time_list[i] :=
stop_time_dev; printf("Computed GCD is: %d.\n\n",res); end do; end
proc)],[proc
() local num_one, num_two, i, res, dev_algo, start_time_res,
stop_time_res,
start_time_dev, stop_time_dev, size_list, bin_time_list,
gcd_time_list, n; n :=
10; size_list := Array(1 .. n); bin_time_list := Array(1 .. n);
gcd_time_list
:= Array(1 .. n); for i to n do num_one := roll(); num_two := roll();
printf(
"***** Testing for numbers: %d and %d ******\n\n",num_one,num_two);
start_time_res := time(); res := bin_gcd(num_one,num_two);
stop_time_res :=
time()-start_time_res; start_time_dev := time(); dev_algo := gcd
(num_one,
num_two); stop_time_dev := time()-start_time_dev; size_list[i] := max
(abs(
num_one),abs(num_two)); bin_time_list[i] := stop_time_res;
gcd_time_list[i] :=
stop_time_dev; printf("Computed GCD is: %d.\n\n",res); end do; end
proc])
```

```
> gcd_test():
```