# Question 2 (Part a and b):

```
> restart:
> `mod` := mods:
> mig_bound := proc(pol::polynom, indeterminate_var)

  local deg_var:
  deg_var := degree(pol, indeterminate_var):
  return 2^deg_var*ceil(sqrt(deg_var+1))*maxnorm(pol):

  end proc:
> diophantine_solver := proc(a, b, c, x, p)

  local g, sigma, tau, q, s, t:
  g := Gcdex(a, b, x, 's', 't') mod p:
  if g <> 1 then
      error "FAIL":
  fi:
  sigma := Rem(c*s, b, x, 'q') mod p:
  tau := expand(c*t+q*a) mod p:

  return(sigma, tau):

  end proc:
> check_mul := proc(pol_a::polynom, pol_b::polynom, pol_c::polynom)

  local a, b, c:
  a, b, c := pol_a, pol_b, pol_c:

  if expand(a*b) = c then
      return 'PASS':
  else
      return 'FAIL':
  fi:

  end proc:
> hensel_lift_algo := proc(pol::polynom, u_in::polynom,
  w_in::polynom, param_p::prime)

  local a, u0, w0, u, w, p, alpha, beta, k, e_k, c_k, s, t:
  a, u0, w0, p, k := pol, u_in, w_in, param_p, 1:
  alpha := lcoeff(a):
```

```
  a := alpha*a:
  beta := mig_bound(expand(a - (u0*w0)), x):
  beta := alpha*beta:
  u := alpha*(u0/lcoeff(u0)) mod p:
  w := alpha*(w0/lcoeff(w0)) mod p:

  while true do:

  e_k := expand(a - (u*w)):
  if e_k = 0 then
      return (primpart(u), primpart(w)):
  fi:
  if p^k > 2*beta then
      return "FAIL":
  fi:

  c_k := e_k/p^k mod p:

  s, t := diophantine_solver(w0, u0, c_k, x, p):
  u, w := u + s*p^k, w + t*p^k:
  u, w := alpha*u/lcoeff(u) mod p^(k+1), alpha*w/lcoeff(w) mod p^
  (k+1):
  k := k+1:

  od:

  end proc:
```

```
> test_a := x^4 - 2*x^3 - 233*x^2 - 214*x + 85:
  test_u_in := x^2 - 3*x - 2:
  test_w_in := x^2 + x + 3:
  test_p := 7:

  printf("\nINPUTS:\n\n1)POLYNOMIAL: %a.\n2)U INITIAL: %a.\n3)W
  INITIAL: %a.\n4)PRIME: %a.\n", test_a, test_u_in, test_w_in,
  test_p):
```

```
INPUTS:

1)POLYNOMIAL: x^4-2*x^3-233*x^2-214*x+85.
2)U INITIAL: x^2-3*x-2.
3)W INITIAL: x^2+x+3.
4)PRIME: 7.
```

```
> u_a, v_a := hensel_lift_algo(test_a, test_u_in, test_w_in, test_p);
```

$$u\_a, v\_a := x^2 - 17\,x + 5, x^2 + 15\,x + 17 \qquad \textbf{(1)}$$

```
> check_mul(u_a, v_a, test_a);
```

*PASS* **(2)**

```
> test_b := 48*x^4 - 22*x^3 + 47*x^2 + 144:
  test_u_in_b := x^2 + 4*x + 2:
  test_w_in_b := x^2 + 4*x + 5:
  test_p_b := 7:

  printf("\nINPUTS:\n\n1)POLYNOMIAL: %a.\n2)U INITIAL: %a.\n3)W
  INITIAL: %a.\n4)PRIME: %a.\n", test_b, test_u_in_b, test_w_in_b,
  test_p_b):
```

```
INPUTS:

1)POLYNOMIAL: 48*x^4-22*x^3+47*x^2+144.
2)U INITIAL: x^2+4*x+2.
3)W INITIAL: x^2+4*x+5.
4)PRIME: 7.
```

```
> u_b, v_b := hensel_lift_algo(expand(6*test_b), test_u_in_b,
  test_w_in_b, test_p_b);
```

$$u\_b, v\_b := 6\,x^2 - 11\,x + 12, 8\,x^2 + 11\,x + 12 \qquad \textbf{(3)}$$

```
> check_mul(u_b, v_b, test_b);
```

*PASS* **(4)**