Question 4:

```
> restart:
> INT := proc(func_f::algebraic, indet_var::name)

  local f, x, u, i, n, v, w, const, dum_var_one, f_one, f_two,
  hold_pow, l_p_one:
  f, x, const := func_f, indet_var, 1:

  (* Integral of a constant int {c} dx = c*x *)

  if diff(f, x) = 0 then
      return (f*x):
  elif type(f, name) then
      if f = x then
          return (x^2 / 2):
      fi:

  (* Integral rule for multiplication *)

  elif op(0, f) = `*` then
      for i from 1 to nops(f) do:
          if diff(op(i, f), x) = 0 then
              const := const * op(i, f):
          fi:
      od:
      u := f / const:

      if const <> 1 then
          return (const*(INT(u, x))):
      elif nops(u) >= 2 then
          f_one := op(1, u):
          f_two := op(2, u):

          (* Integral rule for x^n * exp(x) *)

          if op(1, f_one) = x and degree(f_one) > 1 and op(0, f_two)
  = exp then
              hold_pow := op(2, f_one):
              return (const * (f_one * f_two - INT((hold_pow*x^

              (hold_pow - 1)* f_two), x))):
          elif degree(f_one) = 1 and op(1, f_two) = x then
              return (const * (f_one * f_two - INT(f_two, x))):

          (* Integral rule for x^n * log(x) *)
```

```
        elif op(1, f_one) = x and degree(f_one) > 1 and op(0,
f_two) = ln then
            l_p_one := op(2, f_one):
            return (const * ((x^(l_p_one + 1)/(l_p_one + 1)) * log
(x) - (x^(l_p_one              + 1)/(l_p_one + 1)^2))):
        elif degree(f_one) = 1 and op(0, f_two) = ln then
            return (const * ((x^2/2) * log(x) - (x^(2)/(2)^2))):
        fi:
    fi:
    return (const * INT(u, x)):

(* Integral rule for addition *)

elif op(0, f) = `+` then
    return (add(INT(u, x), u = f)):

(* Integral rule for power *)

elif op(0, f) = `^` then
    if op(1, f) = x then
        n := op(2, f):
        if n <> -1 then
            return ((x^(n+1)) / (n+1)):
        else:
            return log(x):
        fi:
    else:
        return (log(abs(op(1, f))) / lcoeff(op(1, f))):
    fi:

(* Integral rule for exp *)

elif op(0, f) = exp then
    v := op(1, f):
    return exp(v) / diff(v, x):

(* Integral rule for ln *)

elif op(0, f) = ln then
    dum_var_one := op(1, f):
    return (dum_var_one * log(dum_var_one) - dum_var_one):

(* Integral rule for sin(x) *)

elif op(0, f) = sin then
    if diff(op(1, f), x) = 0 then
        return (-cos(x)):
    else:
        w := op(1, f):
        return (diff(w, x) * (-cos(x))):
    fi:

(* Integral rule for cos(x) *)

elif op(0, f) = cos then
    if diff(op(1, f), x) = 0 then
        return (sin(x)):
    else:
        w := op(1, f):
        return (diff(w, x) * (sin(x))):
    fi:
fi:

return 'INT(f(x))':
end proc:
```

```
> f_one := x^2 + 2*x + 1;
  int_f_one := INT(f_one, x);
```

$$f\_one := x^2 + 2\,x + 1$$

$$int\_f\_one := \frac{1}{3}\,x^3 + x^2 + x \tag{1}$$

```
> f_two := x^(-1) + 2*x^(-2) + 3*x^(-1/2);
  int_f_two := INT(f_two, x);
```

$$f\_two := \frac{1}{x} + \frac{2}{x^2} + \frac{3}{\sqrt{x}}$$

$$int\_f\_two := \ln(x) - \frac{2}{x} + 6\,\sqrt{x} \tag{2}$$

```
> f_three := exp(x) + log(x) + sin(x);
  int_f_three := INT(f_three, x);
```

$$f\_three := \mathrm{e}^x + \ln(x) + \sin(x)$$

$$int\_f\_three := \mathrm{e}^x + x\,\ln(x) - x - \cos(x) \tag{3}$$

```
> f_four := 2*f(x) + 3*y*x/2 + 3*ln(2);
  int_f_four := INT(f_four, x);
```

$$f\_four := 2\,f(x) + \frac{3\,y\,x}{2} + 3\,\ln(2)$$

$$int\_f\_four := 2\,INT(f(x)) + \frac{3\,y\,x^2}{4} + 3\,\ln(2)\,x \tag{4}$$

```
> f_five := x^2*exp(x) + 2*x*exp(x);
```

```
  int_f_five := INT(f_five, x);
```

$$f\_five := x^2\,\mathrm{e}^x + 2\,x\,\mathrm{e}^x$$

$$int\_f\_five := x^2\,\mathrm{e}^x \tag{5}$$

```
> f_six := 4*x^3*log(x);
  int_f_six := INT(f_six, x);
```

$$f\_six := 4\,x^3\,\ln(x)$$

$$int\_f\_six := x^4\,\ln(x) - \frac{x^4}{4} \tag{6}$$

```
> f_seven := 2 / (3*x + 1);
  int_f_seven := INT(f_seven, x);
```

$$f\_seven := \frac{2}{3\,x + 1}$$

$$int\_f\_seven := \frac{2\,\ln(|3\,x + 1|)}{3} \tag{7}$$