```
> restart:
```

Question 2C:

```
> M := 'M':
  rec_eq := {M(k) = 2*M(k-1) + 2^(k-1), M(0)=0};
  rec_soln := rsolve(rec_eq, M(k));
```

$$rec\_eq := \{M(0) = 0, M(k) = 2\,M(k-1) + 2^{k-1}\}$$

$$rec\_soln := -\frac{2^k}{2} + \left(\frac{k}{2} + \frac{1}{2}\right) 2^k \tag{1}$$

```
> new_M := subs(k=log[2](n),rec_soln):
  M_n := simplify(new_M);
```

$$M\_n := \frac{n \ln(n)}{2 \ln(2)} \tag{2}$$

Question 2D:

```
> d_FFT := proc(param_n::posint, param_arr::anything,
  param_p::posint, param_w::anything)

          local n, A, B, C, eval_B, eval_C, Y, T, i, j, w, p:
          n, A, w, p := param_n, param_arr, param_w, param_p:

          if n = 1 then
              return A;
          fi:

          B := [seq(A[2*i+1], i=0..((n/2)-1))]:
          C := [seq(A[2*i+2], i=0..((n/2)-1))]:

          eval_B := d_FFT((n/2), B, p, w^2 mod p):
          eval_C := d_FFT((n/2), C, p, w^2 mod p):

          Y := 1:
          for j from 1 to (n/2) do:
              T := Y*eval_C[j] mod p:
              A[j] := (eval_B[j] + T) mod p:
              A[j+(n/2)] := (eval_B[j] - T) mod p:
              Y := w*Y mod p:
          od:
          return A;
          end proc:

> printf("\nInputs:\n"):
  with(NumberTheory):
```

```
A := [1,2,3,4,3,2,1,0];
n := nops(A);
p := 97;
alpha := PrimitiveRoot(p):
omega := alpha^12 mod p;
```

```
Inputs:
```

$$A := [1, 2, 3, 4, 3, 2, 1, 0]$$
$$n := 8$$
$$p := 97$$
$$\omega := 64 \tag{3}$$

```
> printf("\nThe discrete FFT of A:\n"):
  FFT_A := d_FFT(n, A, p, omega);
```

```
The discrete FFT of A:
```

$$FFT\_A := [16, 48, 0, 16, 0, 36, 0, 86] \tag{4}$$

```
> omega_inv := omega^(-1) mod p;
  printf("\nThe inverse discrete FFT of A:\n"):
  inv_FFT_A := d_FFT(n, FFT_A, p, omega_inv);

  manual_comp := [1,2,3,4,3,2,1,0]:
  for i from 1 to nops(manual_comp) do:
      manual_comp[i] := 8*manual_comp[i] mod p:
  od:
  manual_comp := manual_comp;

  printf("\nThe above calculation shows that FFT(n, B, p, w^-1) = n*A
  mod p.\n"):
```

$$omega\_inv := 47$$

```
The inverse discrete FFT of A:
```

$$inv\_FFT\_A := [8, 16, 24, 32, 24, 16, 8, 0]$$
$$manual\_comp := [8, 16, 24, 32, 24, 16, 8, 0]$$

```
The above calculation shows that FFT(n, B, p, w^-1) = n*A mod p.
```

Question 2E:

```
> printf("\nInput Polynomials:\n"):
  pol_a := -x^3 + 3*x + 1;
```

```
pol_b := 2*x^4 - 3*x^3 - 2*x^2 + x + 1;
```

Input Polynomials:

$$pol\_a := -x^3 + 3x + 1$$
$$pol\_b := 2x^4 - 3x^3 - 2x^2 + x + 1 \quad (5)$$

```
> deg_a := degree(pol_a);
  deg_b := degree(pol_b);
  total_deg := deg_a + deg_b;
```

$$deg\_a := 3$$
$$deg\_b := 4$$
$$total\_deg := 7 \quad (6)$$

```
> printf("\nThe coeffs. of A mod 97 and B mod 97 (without padding).
  \n"):

  dum_A := [coeffs(pol_a)]:
  for i from 1 to nops(dum_A) do
      dum_A[i] := dum_A[i] mod 97:
  od:
  dum_A := dum_A;
  dum_B := [coeffs(pol_b)]:
  for i from 1 to nops(dum_B) do
      dum_B[i] := dum_B[i] mod 97:
  od:
  dum_B := dum_B;
```

The coeffs. of A mod 97 and B mod 97 (without padding).

$$dum\_A := [96, 3, 1]$$
$$dum\_B := [2, 94, 95, 1, 1] \quad (7)$$

```
> printf("\nThe coeffs. of A mod 97 and B mod 97 (with padding).\n"):

  dum_A := [1, 3, 0, 96, 0, 0, 0, 0];
  dum_B := [1, 1, 95, 94, 2, 0, 0, 0];
```

The coeffs. of A mod 97 and B mod 97 (with padding).

$$dum\_A := [1, 3, 0, 96, 0, 0, 0, 0]$$
$$dum\_B := [1, 1, 95, 94, 2, 0, 0, 0] \quad (8)$$

```
> printf("\nThe discrete FFT of A:\n"):
  FFT_A := d_FFT(n, dum_A, p, omega);
```

The discrete FFT of A:

$$FFT\_A := [3, 46, 89, 87, 96, 53, 10, 12] \quad (9)$$

```
> printf("\nThe discrete FFT of B:\n"):
  FFT_B := d_FFT(n, dum_B, p, omega);
```

The discrete FFT of B:

$$FFT\_B := [96, 63, 93, 95, 3, 41, 14, 88] \tag{10}$$

```
> printf("\nThe discrete FFT of C:\n"):
  FFT_C := [seq((FFT_A[i]*FFT_B[i]) mod p, i = 1..n)];
```

The discrete FFT of C:

$$FFT\_C := [94, 85, 32, 20, 94, 39, 43, 86] \tag{11}$$

```
> printf("\nThe inverse discrete FFT of C:\n"):
  inv_FFT_C := d_FFT(n, FFT_C, p, omega_inv);
```

The inverse discrete FFT of C:

$$inv\_FFT\_C := [8, 32, 8, 17, 33, 64, 24, 81] \tag{12}$$

```
> inv_n := (n^(p-2)) mod p;
  recovered_coeff_C := [seq((inv_FFT_C[i]*inv_n) mod p, i = 1..n)];
  manual_prod := expand(pol_a * pol_b) mod p;
  printf("\nThe recovered coeff's for C match the expanded product of
  polynomials A and B."\n):
```

$$inv\_n := 85$$

$$recovered\_coeff\_C := [1, 4, 1, 87, 89, 8, 3, 95]$$

$$manual\_prod := 95\,x^7 + 3\,x^6 + 8\,x^5 + 89\,x^4 + 87\,x^3 + x^2 + 4\,x + 1$$

The recovered coeff's for C match the expanded product of polynomials A and B.