| galois |

Exercise 1:

The following adds integers mod $(2^{31}-1)$:

```
plus : ([31],[31]) -> [31]
plus (a, b) = if sab @ 0 then sab' + 1 else sab'
  where
   sab = ((zero:[1]) # a) + ((zero:[1]) # b)  // sab is 32 bits
   sab' = drop `{1} sab
```

Note: sab is a mod (2^{31}) sum when the leading bit is removed as in sab'. If sab != 0 then the mod (2^{31}) sum sab' is the same as the mod $(2^{31}-1)$ sum. If sab == 0 the mod (2^{31}) sum is 1 past the mod $(2^{31}-1)$ sum so sab' should be 1, hence the if statement in plus.

Exercise 2:

Adding a sequence of numbers mod $(2^{31}-1)$ is accomplished with this:

```
add xs = sums ! 0
  where
    sums = [0] # [plus (s,x) | s <- sums | x <- xs]
```

where plus is the function of Exercise 1 that adds mod $(2^{31}-1)$.

Exercise 3:

Write the function v ss that outputs:

$(S_{15}<<<_{31}15)+(S_{13}<<<_{31}17)+(S_{10}<<<_{31}21)+(S_4<<<_{31}20)+(S_0<<<_{31}8)+S_0 \bmod (2^{31}-1)$

The output of v is 31 bits wide. The addition is mod $(2^{31}-1)$.

Taking the sequence $S_{15}...S_0$ as input ss:

```
v (ss) = add [ s <<< c | s <- ss@@[15,13,10,4,0,0]
                       | c <- [15,17,21,20,8,0]]
```

Exercise 4:

```
LFSRWithInitializationMode : ([31], [16][31]) -> [16][31]
LFSRWithInitializationMode (u,ss) = ss @@ [1 .. 15] # [s16]
  where
    vu = add [v(ss), u]
    s16 = if vu == 0 then `0x7FFFFFFF else vu
```

Exercise 5:

```
LFSRWithWorkMode : ([16][31]) -> [16][31]
LFSRWithWorkMode (ss) = ss @@ [1 .. 15] # [s16]
  where
    vu = v(ss)
    s16 = if vu == 0 then `0x7FFFFFFF else vu
```

Exercise 6:

```
BitReorganization : [16][31] -> [4][32]
BitReorganization ss =
   [hi(s15)#lo(s14), lo(s11)#hi(s9), lo(s7)#hi(s5), lo(s2)#hi(s0)]
  where
    lo(x) = x @@ [15 .. 30]
    hi(x) = x @@ [0  .. 15]
    [s0,s2,s5,s7,s9,s11,s14,s15] = ss @@ [0,2,5,7,9,11,14,15]
```

Exercise 7:

```
S X = Y0 # Y1 # Y2 # Y3
  where
    [X0, X1, X2, X3] = split X
    [Y0, Y1, Y2, Y3] = [S0(X0), S1(X1), S2(X2), S3(X3)]
S0(x) = S0Box @ x
S1(x) = S1Box @ x
S2 = S0
S3 = S1
```

Exercise 8:

```
L1(X) = X ^ X <<< 2 ^ X <<< 10 ^ X <<< 18 ^ X <<< 24
L2(X) = X ^ X <<< 8 ^ X <<< 14 ^ X <<< 22 ^ X <<< 30
```

Exercise 9:

```
a) W = (X0 ^ R1) + R2
b) W1 = R1 + X1
   W2 = R2 ^ X2
c) [W1H, W1L] = split W1
   [W2H, W2L] = split W2
   R1 = S (L1 (W1L # W2H))
   R2 = S (L2 (W2L # W1H))
d) F ([X0, X1, X2], [R1, R2]) = (W, [R1', R2'])
     where
       W = (X0 ^ R1) + R2
       W1 = R1 + X1
       W2 = R2 ^ X2
       [W1H, W1L] = split W1
       [W2H, W2L] = split W2
       R1' = S (L1 (W1L # W2H))
       R2' = S (L2 (W2L # W1H))
```

Example 10:

```
LoadKey (key, iv) = [k # d # i | k <- ks | i <- is | d <- ds]
  where
    ks = split key
    is = split iv
    ds = [0b100010011010111, 0b0100111010111100, 0b1100010011010111,
          0b0010011010111110, 0b1010111110001001, 0b0111010111100010,
          0b1110001001101101, 0b0001001101011111, 0b1001101011110000,
          0b0101111100010011, 0b1101011111000100, 0b0011010111110001,
          0b1011111000100110, 0b0111100010011010, 0b1111000100110100]
```