Exercise 1:

Salsa20.cry:

```
quarterround : [4][32] -> [4][32]
quarterround [y0, y1, y2, y3] = [z0, z1, z2, z3]
  where
    z1 = y1 ^ ((y0 + y3) <<< 0x7)
    z2 = y2 ^ ((z1 + y0) <<< 0x9)
    z3 = y3 ^ ((z2 + z1) <<< 0xd)
    z0 = y0 ^ ((z3 + z2) <<< 0x12)

rowround : [16][32] -> [16][32]
rowround [y0,y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,y15] =
         [z0,z1,z2,z3,z4,z5,z6,z7,z8,z9,z10,z11,z12,z13,z14,z15]
  where
    [z0,z1,z2,z3]     = quarterround [y0,y1,y2,y3]
    [z5,z6,z7,z4]     = quarterround [y5,y6,y7,y4]
    [z10,z11,z8,z9]   = quarterround [y10,y11,y8,y9]
    [z15,z12,z13,z14] = quarterround [y15,y12,y13,y14]

rowround_opt : [16][32] -> [16][32]
rowround_opt ys = join [(quarterround (yi<<<i))>>>i
                       | yi <- split ys | i <- [0 .. 3]]

columnround : [16][32] -> [16][32]
columnround [x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15] =
            [y0,y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,y15]
  where
    [y0,y4,y8,12]    = quarterround [x0,x4,x8,x12]
    [y5,y9,y13,y1]   = quarterround [x5,x9,x13,x1]
    [y10,y14,y2,y6]  = quarterround [x10,x14,x2,x6]
    [y15,y3,y7,y11]  = quarterround [x15,x3,x7,x11]

columnround_opt : [16][32] -> [16][32]
columnround_opt xs = join (transpose [ (quarterround (xi<<<i))>>>i
                                     | xi <- transpose(split xs)
                                     | i <- [0 .. 3] ])

doubleround : [16][32] -> [16][32]
doubleround(xs) = rowround(columnround(xs))

littleendian : [4][8] -> [32]
littleendian b = join(reverse b)

littleendian_inverse : [32] -> [4][8]
littleendian_inverse b = reverse(split b)

Salsa20 : [64][8] -> [64][8]
Salsa20 xs = join ar
  where
    ar = [ littleendian_inverse words | words <- xw + zs@10 ]
    xw = [ littleendian xi | xi <- split xs ]
    zs = [xw] # [ doubleround zi | zi <- zs ]

// Salsa 20 supports two key sizes, [16][8] and [32][8]
Salsa20_expansion : {a} (a >= 1, 2 >= a) => ([16*a][8], [16][8]) -> [64][8]
Salsa20_expansion(k, n) = z
```

```
    where
      [s0, s1, s2, s3] = split "expand 32-byte k" : [4][4][8]
      [t0, t1, t2, t3] = split "expand 16-byte k" : [4][4][8]
      x = if(`a == 2) then s0 # k0 # s1 # n # s2 # k1 # s3
                      else t0 # k0 # t1 # n # t2 # k0 # t3
      z = Salsa20(x)
      [k0, k1] = (split(k#zero)):[2][16][8]

  Salsa20_encrypt:{a,l} (a>=1,2>=a,l<=2^^70) => ([16*a][8],[8][8],[l][8])->[l][8]
  Salsa20_encrypt(k, v, m) = c
    where
      salsa = take (join [ Salsa20_expansion(k, v#(reverse (split i)))
                         | i <- [0, 1 ... ] ])
      c = m ^ salsa
```

```
quarterround [0xd3917c5b, 0x55f1c407, 0x52a58a7a, 0x8f887a3b] =
             [0x3e2f308c, 0xd90a8f36, 0x6ab2a923, 0x2883524c]

rowround [0x08521bd6, 0x1fe88837, 0xbb2aa576, 0x3aa26365,
          0xc54c6a5b, 0x2fc74c2f, 0x6dd39cc3, 0xda0a64f6,
          0x90a2f23d, 0x067f95a6, 0x06b35f61, 0x41e4732e,
          0xe859c100, 0xea4d84b7, 0x0f619bff, 0xbc6e965a] =
         [0xa890d39d, 0x65d71596, 0xe9487daa, 0xc8ca6a86,
          0x949d2192, 0x764b7754, 0xe408d9b9, 0x7a41b4d1,
          0x3402e183, 0x3c3af432, 0x50669f96, 0xd89ef0a8,
          0x0040ede5, 0xb545fbce, 0xd257ed4f, 0x1818882d]

columnround [0x08521bd6, 0x1fe88837, 0xbb2aa576, 0x3aa26365,
             0xc54c6a5b, 0x2fc74c2f, 0x6dd39cc3, 0xda0a64f6,
             0x90a2f23d, 0x067f95a6, 0x06b35f61, 0x41e4732e,
             0xe859c100, 0xea4d84b7, 0x0f619bff, 0xbc6e965a] =
            [0x8c9d190a, 0xce8e4c90, 0x1ef8e9d3, 0x1326a71a,
             0x90a20123, 0xead3c4f3, 0x63a091a0, 0xf0708d69,
             0x789b010c, 0xd195a681, 0xeb7d5504, 0xa774135c,
             0x481c2027, 0x53a8e4b5, 0x4c1f89c5, 0x3f78c9c8]

littleendian [86, 75, 30, 9] = 0x091e4b56

Salsa20 [88,118,104,54,79,201,235,79,3,81,156,47,203,26,244,243,
         191,187,234,136,211,159,13,115,76,55,82,183,3,117,222,37,
         86,16,179,207,49,237,179,48,1,106,178,219,175,199,166,48,
         238,55,204,36,31,240,32,63,15,83,93,161,116,147,48,113] =
        [0xb3, 0x13, 0x30, 0xca, 0xdb, 0xec, 0xe8, 0x87,
         0x6f, 0x9b, 0x6e, 0x12, 0x18, 0xe8, 0x5f, 0x9e,
         0x1a, 0x6e, 0xaa, 0x9a, 0x6d, 0x2a, 0xb2, 0xa8,
         0x9c, 0xf0, 0xf8, 0xee, 0xa8, 0xc4, 0xbe, 0xcb,
         0x45, 0x90, 0x33, 0x39, 0x1d, 0x1d, 0x96, 0x1a,
         0x96, 0x1e, 0xeb, 0xf9, 0xbe, 0xa3, 0xfb, 0x30,
         0x1b, 0x6f, 0x72, 0x72, 0x76, 0x28, 0x98, 0x9d,
         0xb4, 0x39, 0x1b, 0x5e, 0x6b, 0x2a, 0xec, 0x23]

Salsa20_encrypt ([0x23,0x12,0x14,0x72,0xEE,0xEa,0x45,0x23,
                   0x4A,0x2A,0x6D,0x55,0xF2,0xCC,0xCA,0xC2],
                  [0x11,0x78,0x8E,0x3B,0x77,0x63,0x3A,0x3C],
                  [0xDD,0x34,0x67,0x33,0x23,0xC4,0xD3,0xEE]) =
                  [0x21,0xc1,0x66,0xcb,0x24,0x58,0x7e,0x34]
```

```
Salsa20_encrypt ([0x23,0x12,0x14,0x72,0xEE,0xEa,0x45,0x23,
                  0x4A,0x2A,0x6D,0x55,0xF2,0xCC,0xCA,0xC2],
                 [0x11,0x78,0x8E,0x3B,0x77,0x63,0x3A,0x3C],
                 [0x21,0xC1,0x66,0xCB,0x24,0x58,0x7E,0x34]) =
                 [0xDD,0x34,0x67,0x33,0x23,0xC4,0xD3,0xEE]
```

Exercise 2:

Prove Salsa20 encrypts and decrypts correctly:

```
Salsa20_can_encrypt_and_decrypt : [32][8] -> [8][8] -> [200][8] -> Bit
property Salsa20_can_encrypt_and_decrypt k v m =
   Salsa20_encrypt(k,v,Salsa20_encrypt(k,v,m)) == m
Main> :prove Salsa20_can_encrypt_and_decrypt
Q.E.D.
(Total Elapsed Time: 1.024s, using "Z3")
```

Prove if `x1` ≠ `x2` then `doubleround x1` ≠ `doubleround x2` :

```
property Salsa20_has_no_collisions x1 x2 =
   if(x1 != x2) then (doubleround x1) != (doubleround x2) else True
Main> :prove Salsa20_has_no_collisions
Q.E.D.
(Total Elapsed Time: 46.972s, using "Z3")
```

Prove `columnround` is the transpose of `rowround`:

```
property columnround_is_transpose_of_rowround ys =
   rowround ys == join(transpose(split`{4}(columnround xs)))
   where xs = join(transpose(split`{4} ys))
Main> :prove columnround_is_transpose_of_rowround
Q.E.D.
(Total Elapsed Time: 0.014s, using "Z3")
```

Prove `rowround` x == `rowround_opt` x:

```
property rowround_opt_is_rowround x = rowround x == rowround_opt x
Main> :prove rowround_opt_is_rowround
Q.E.D.
(Total Elapsed Time: 0.017s, using "Z3")
```

Prove `littleendian` is invertible:

```
property littleendian_is_invertable b =
   littleendian_inverse(littleendian b) == b
Main> :prove littleendian_is_invertable
Q.E.D.
(Total Elapsed Time: 0.010s, using "Z3")
```

Exercise 3:

s1.saw:

```
import "Salsa20.cry";

let alloc_init ty v = do {
   p <- llvm_alloc ty;
   llvm_points_to p (llvm_term v);
   return p;
};

let quarterround_setup = do {
   y0 <- llvm_fresh_var "y0" (llvm_int 32);
   y1 <- llvm_fresh_var "y1" (llvm_int 32);
   y2 <- llvm_fresh_var "y2" (llvm_int 32);
   y3 <- llvm_fresh_var "y3" (llvm_int 32);
   p0 <- alloc_init (llvm_int 32) {{ y0 }};
   p1 <- alloc_init (llvm_int 32) {{ y1 }};
   p2 <- alloc_init (llvm_int 32) {{ y2 }};
   p3 <- alloc_init (llvm_int 32) {{ y3 }};

   llvm_execute_func [p0, p1, p2, p3];

   let zs = {{ quarterround [y0,y1,y2,y3] }};
   llvm_points_to p0 (llvm_term {{ zs@0 }});
   llvm_points_to p1 (llvm_term {{ zs@1 }});
   llvm_points_to p2 (llvm_term {{ zs@2 }});
   llvm_points_to p3 (llvm_term {{ zs@3 }});
};

let main = do {
   mm <- llvm_load_module "salsa20.bc";
   qr <- llvm_verify mm "s20_quarterround" [] false quarterround_setup yices;
   print "Done!";
};
```

Run `saw s1.saw`:

```
[12:00:43.266] Verifying s20_quarterround ...
[12:00:43.267] Simulating s20_quarterround ...
[12:00:43.273] Checking proof obligations s20_quarterround ...
[12:00:43.303] Proof succeeded! s20_quarterround
[12:00:43.303] Done!
```

Exercise 4:

May not finish

Exercise 5:

```
[13:18:50.749] Verifying s20_quarterround ...
[13:18:50.750] Simulating s20_quarterround ...
[13:18:50.756] Checking proof obligations s20_quarterround ...
[13:18:50.782] Proof succeeded! s20_quarterround
[13:18:50.823] Verifying s20_rowround ...
[13:18:50.823] Simulating s20_rowround ...
[13:18:50.833] Checking proof obligations s20_rowround ...
[13:18:50.851] Proof succeeded! s20_rowround
```

```
[13:18:50.889] Verifying s20_columnround ...
[13:18:50.890] Simulating s20_columnround ...
[13:18:50.899] Checking proof obligations s20_columnround ...
[13:18:50.918] Proof succeeded! s20_columnround
[13:18:50.955] Verifying s20_doubleround ...
[13:18:50.956] Simulating s20_doubleround ...
[13:18:50.974] Checking proof obligations s20_doubleround ...
[13:18:54.593] Proof succeeded! s20_doubleround
[13:18:54.632] Verifying s20_hash ...
[13:18:54.633] Simulating s20_hash ...
[13:18:54.633] Registering overrides for `s20_doubleround`
[13:18:54.633]   variant `Symbol "s20_doubleround"`
[13:18:54.647] Matching 1 overrides of  s20_doubleround ...
[13:18:54.649] Branching on 1 override variants of s20_doubleround ...
[13:18:54.651] Applied override! s20_doubleround
[13:18:54.652] Matching 1 overrides of  s20_doubleround ...
[13:18:54.652] Branching on 1 override variants of s20_doubleround ...
[13:18:54.653] Applied override! s20_doubleround
[13:18:54.654] Matching 1 overrides of  s20_doubleround ...
[13:18:54.654] Branching on 1 override variants of s20_doubleround ...
[13:18:54.655] Applied override! s20_doubleround
[13:18:54.655] Matching 1 overrides of  s20_doubleround ...
[13:18:54.655] Branching on 1 override variants of s20_doubleround ...
[13:18:54.656] Applied override! s20_doubleround
[13:18:54.657] Matching 1 overrides of  s20_doubleround ...
[13:18:54.657] Branching on 1 override variants of s20_doubleround ...
[13:18:54.658] Applied override! s20_doubleround
[13:18:54.658] Matching 1 overrides of  s20_doubleround ...
[13:18:54.658] Branching on 1 override variants of s20_doubleround ...
[13:18:54.659] Applied override! s20_doubleround
[13:18:54.659] Matching 1 overrides of  s20_doubleround ...
[13:18:54.660] Branching on 1 override variants of s20_doubleround ...
[13:18:54.661] Applied override! s20_doubleround
[13:18:54.661] Matching 1 overrides of  s20_doubleround ...
[13:18:54.661] Branching on 1 override variants of s20_doubleround ...
[13:18:54.662] Applied override! s20_doubleround
[13:18:54.662] Matching 1 overrides of  s20_doubleround ...
[13:18:54.662] Branching on 1 override variants of s20_doubleround ...
[13:18:54.663] Applied override! s20_doubleround
[13:18:54.664] Matching 1 overrides of  s20_doubleround ...
[13:18:54.664] Branching on 1 override variants of s20_doubleround ...
[13:18:54.665] Applied override! s20_doubleround
[13:18:54.688] Checking proof obligations s20_hash ...
[13:18:55.181] Proof succeeded! s20_hash
[13:18:55.182] Done!
```