1. Converting floating point to binary
2. 5.75
   * Start by converting 5 to binary using repeated division to get the binary result of 5.
     1. 5/2 = 2 with 1 left over
     2. 2/2 = 1 with 0 left over
     3. ½ = 0 with 1 leftover
   * Write out remainder in reverse order: 101
   * Then take the decimal and multiply by 2 until you get fractional parts equal to 0
     1. .75x2 = 1 + .5
     2. .5 x 2 = 1 + 0
     3. So .75 converted into binary is .11
   * **Now adding both binary results together gives: 101.11**
3. 63/64 = .984375 in decimal
   * Since there are no whole numbers, we have nothing before the decimal point
   * Same steps as above
     1. .984375 x 2 = 1 + .96875
     2. .96875 x 2 = 1 + .9375
     3. .9375 x 2 = 1 + .875
     4. .875 x 2 = 1 + .75
     5. .75 x 2 = 1 + .5
     6. .5 x 2 = 1 + 0
   * **This becomes 0.111111**
4. 9.8125
   * First we convert 9 to binary using repeated division
     1. 9/2 = 4 with 1 leftover
     2. 4/2 = 2 with 0 leftover
     3. 2/2 = 1 with 0 leftover
     4. ½ = 0 with 1 leftover
   * 9 = 1001
   * Next take .8125 and multiply by 2
     1. .8125 x 2 = 1 + .625
     2. .625 x 2 = 1 + .25
     3. .25 x 2 = 0 + .5
     4. .5 x 2 = 1 + 0
   * **Then add them together and you get: 1001.1101**
5. Convert 34.890625 into IEEE-754 floating point representation
   * First convert both the whole and decimal parts to binary
     1. 34.890625 becomes 100010.0111110110
     2. Move the decimal 5 places becomes 1.000100111110110x2^5
     3. The sign bit is 0 since the number is positive
     4. For single precision add base 127 and the exponent of 5 resulting in 132
     5. Use repeated division on 132 to get it’s binary result of 10000100
     6. Now we have 010000100 for the mantissa
     7. Now we combine all parts together
   * Final result is: 01000010010001001111101100000000
6. To convert 00111101100000000000000000000000 to decimal
   * Reverse the above steps to identify the mantisa, exponent, and sign bit
   * Then apply the fraction and whole numbers as a decimal
   * **Final result is 0.0625**
7. A denormalized number is a subnormal number with a floating point decimal. It includes any numbers with a smaller magnitude than 1
   * The largest denormalized number is 127 and the smallest is -126 in 8-bit representation
   * 127 = 01000010111111100000000000000000
   * -126 = 11000010111111000000000000000000