Question 3.2–1: (Solution, p 2) Represent $-300_{(10)}$ and $-1_{(10)}$ in twelve bits using sign-magnitude representation.

Question 3.2–2: (Solution, p 2) Represent $-300_{(10)}$ and $-1_{(10)}$ in twelve bits using two's-complement representation.

Question 3.2–3: (Solution, p 2) What is the smallest (most negative) number you can represent in twelve bits using sign-magnitude representation? In two's-complement representation? Give both the bit pattern of the number and its decimal translation.

Question 3.3–1: (Solution, p 2) Convert each of the following numbers to the 8-bit floating-point representation we saw in class.

a. $2.5_{(10)}$

b. $0.0625_{(10)}$

Question 3.3–2: (Solution, p 2) What decimal number does each of the 8-bit floating-point bit patterns represent?

a. 0 0110 101

b. 1 1011 001

Question 3.3–3: (Solution, p 2) For the following questions, we define an nine-bit floating point number to include one sign bit, five exponent bits (using excess 15), and three mantissa bits. The all-zeroes exponent is reserved for denormalized numbers, and the all-ones exponent is reserved for the special cases.

a. Represent each of the following decimal numbers in this nine-bit system: $1_{(10)}$, $-6_{(10)}$.

b. Convert each of the following nine-bit floating-point patterns into base 10. Your answer should include at least two significant digits.

0 10000 010
1 10011 110

c. Assuming that the system uses the special-case numbers (infinity, negative infinite, not-a-number), what is the largest numerical value you can represent in this nine-bit system? Express your answer in decimal and justify your answer.

Question 3.4–1: (Solution, p 2) Suppose a digital camera uses a 40 MB disk to store pictures. How many 3×5-inch photographs can it store in 24-bit color at the standard printer-quality resolution of 300 pixels per inch?

Question 3.4–2: (Solution, p 2) We examined a compression technique called *run-length encoding* for black-and-white images, in which each byte includes four bits saying how many adjacent black pixels there are and four bits saying how many adjacent white pixels there are. As we saw, this compression technique sometimes actually *expands* a picture. What is the maximum possible expansion factor?

Solution 3.2–1: (Question, p 1) 1001 0010 1100 is $-300_{(10)}$ and 1000 0000 0001 is $-1_{(10)}$.

Solution 3.2–2: (Question, p 1) 1110 1101 0100 is $-300_{(10)}$ and 1111 1111 1111 is $-1_{(10)}$.

Solution 3.2–3: (Question, p 1)

Sign-magnitude: 1111 1111 1111 represents $-2{,}047_{(10)}$

Two's-complement: 1000 0000 0000 represents $-2{,}048_{(10)}$

Solution 3.3–1: (Question, p 1)

a. 0 1000 010

b. 0 0011 000

Solution 3.3–2: (Question, p 1)

a. $0.8125_{(10)}$

b. $-18_{(10)}$

Solution 3.3–3: (Question, p 1)

a.

work	solution
$1 = 1.0 \times 2^0$.	0 01111 000
$-6 = -1.5 \times 2^2 = (1 + 4/8) \times 2^2$.	1 10001 100

b.

problem	solution
0 10000 010	$1.010_{(2)} \times 2^1 = 10.1_{(2)} = 2.5_{(10)}$
1 10011 110	$-1.110_{(2)} \times 2^4 = -11100_{(2)} = -28_{(10)}$

c. To get the largest number, the exponent bits will be 11110 (11111 being reserved for the non-numbers) and the mantissa bits will be 111. $1.111_{(2)} \times 2^{15} = 1111\,0000\,00000000_{(2)} = 61{,}440_{(10)}$

Solution 3.4–1: (Question, p 1)

$$\frac{3 \times 5 \text{ sq. inches}}{\text{picture}} \times \frac{300 \times 300 \text{ pixels}}{\text{sq. inch}} \times \frac{3 \text{ bytes}}{\text{pixel}} \times \frac{\text{MB}}{2^{20} \text{ bytes}} \times = \frac{3.86 \text{ MB}}{\text{picture}}$$

$$\frac{40 \text{ MB}}{\text{disk}} \times \frac{\text{picture}}{3.86 \text{ MB}} \times = \frac{10.4 \text{ pictures}}{\text{disk}}$$

(Of course, since it wouldn't store a fraction of a picture, It would only hold 10 pictures.)

Solution 3.4–2: (Question, p 1) A checkboard pattern, where every other pixel is black, would give the maximum expansion factor. For such a system, each run would be only one pixel long, and the compression system would use four bits to encode each run, for an expansion factor of four.