Problem 5.1 (Read the bulleted list describing rounding modes on page 116 first.)**:** Write down the IEEE double-precision representation for the following decimal numbers,

1. 1.5: Using round up,

   Solution:
   Using double precision, the number $1.5 = 1.100 \times 2^0$ is represented by, **Console:**

```
>> [sign_bit, Exponent_bits, mantissa_bits] = showfloat(1.5)

sign_bit =

    "0"


Exponent_bits =

    "01111111111"


mantissa_bits =

    "1000000000000000000000000000000000000000000000000000"
```

   Note since .5 is a power of two we know that this answer is exact, therefore round down returns the same.

2. 5.1: Using round to nearest,

   Solution:
   Representing 5.1 in base 2 we get, $101.00\overline{1100}$ Using double precision, the number $5.1 = 1.0100\overline{0110} \times 2^2$ is represented by, **Console:**

```
sign_bit =

    "0"


Exponent_bits =

    "10000000001"


mantissa_bits =
```

```
"0100011001100110011001100110011001100110011001100110011001100110"
```

Note that with higher precision the next bit would be a 0 therefore rounding to the nearest will produce the same result.

3. -5.1: Using round towards 0,

Solution:
Using double precision, the number $-5.1 = -1.0100\overline{0110} \times 2^2$ is represented by,
Console:

```
sign_bit =

    "1"


Exponent_bits =

    "10000000001"


mantissa_bits =

    "0100011001100110011001100110011001100110011001100110011001100110"
```

Note that rounding to zero a negative number is equivalent to rounding down the corresponding positive, therefore we would produce the same result as the previous problem, for the same reason

4. -5.1: Using round to down,

Solution:
Using double precision, the number $-5.1 = -1.0100\overline{0110} \times 2^2$ is represented by,
Console:

```
sign_bit =

    "1"


Exponent_bits =

    "10000000001"
```

```
mantissa_bits =

    "0100011001100110011001100110011001100110011001100110011001100110"
```

Rounding to down a negative number is equivalent of rounding up the corresponding positive number, therefore we raise the last digit by adding $(2^{-}52)_2$.

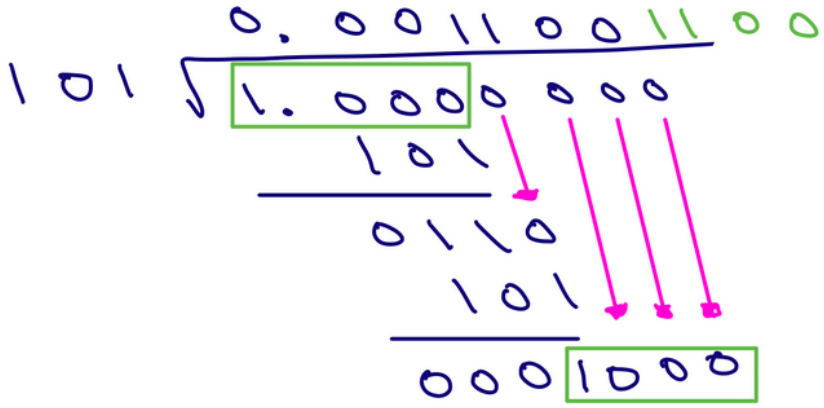
```
sign_bit =

    "1"


Exponent_bits =

    "10000000001"


mantissa_bits =

    "0100011001100110011001100110011001100110011001100110011001100111"
```

Problem 5.2: For full credit you should correctly justify via a hand computation how you arrived at this answer. Write down the IEEE double precision representation for the decimal number 50.2, round to nearest.

Solution:
First we start by finding the base 2 representation for the number 50.2. Note that,

$$2^5 + 2^4 + 2^1 = 32 + 16 + 2 = 50 = (110010)_2.$$

Now we must find the base 2 representation of $.2 = \frac{1}{5}$. With some binary long division we get,



Therefore the binary representation of 50.2 is,

$$(50.2)_{10} = (110010.00\overline{1100})_2.$$

To get the IEEE double precision representation we need to calculate the sign, mantissa and exponent bits. The sign bit is easy, since 50.2 is a positive number we keep the sign bit set to 0. Since our mantissa has to be of the form,

$$1.b_1b_2b_3...b_{52}$$

We know that the decimal will be displaced by 5 digits so therefore the exponent bit will be,

$$(5 + 1023)_{10} = (1028)_{10} = (10000000100)_2$$

We know what the mantissa bits will be through our binary division, and since we displaced the decimal five digits, the mantissa will look like the following until it has reached 52 digits (53 if you count the implicit 1),

$$(1.10010001\overline{1100})_2$$

Expanding out to 52 digits and restating the exponents and sign bits we get,

sign_bit =

"0"

Exponent_bits =

"10000000100"

```
mantissa_bits =

    "1001000110011001100110011001100110011001100110011001100110011001"
```

Following the pattern we know that the next bit will be a 1 so rounding to the nearest will result in a mantissa of,

```
mantissa_bits =

    "1001000110011001100110011001100110011001100110011001100110011010"
```

Luckily rounding to nearest was not enough to add another digit and therefore the exponent bits remain unchanged.

Problem 5.3: What is the gap between 2 and the next larger double-precision number.

Solution:
From class we know that the gap between n and the next number will be,

$$2^E \epsilon.$$

Where E is the actual exponent for n. Since $(2)_{10} = (10)_2$ therefore the actual exponent value is, has to be 1 in order to get it in the form,

$$1.b_1 b_2 b_3 ... b_{52}.$$

Substituting $\epsilon = 2^{-52}$ since that is machine epsilon for IEEE double precision, we get that the gap between 2 and the next larger double-precision number is,

$$2^1 2^{-52} = 2^{-51}.$$

Problem 5.4: What is the gap between 201 and the next larger double precision number.

Solution:
Using the same formula as the last problem, all we need to do is calculate the base 2 representation for 201,

$$201 = 128 + 64 + 8 + 1 = 2^7 + 2^6 + 2^3 + 2^0 = (11001001)_2.$$

Note that our actual exponent would be $E = 7$, thus

$$2^7 2^{-52} = 2^{-45}.$$

Problem 5.5: How many normalized double-precision numbers are their? Express your answer using powers of 2.

Solution:
Consider that all double-precision numbers are represented by 1 sign bit, 11 exponent bits and 52 mantissa bits. Therefore the total representations is calculated by,

$$2 * 2^{11} * 2^{52}.$$

However there are two special cases that are not defined as normalized, and those are produced when our exponent bits are all zero, for our two zeroes and subnormal numbers, and the case where our exponent bits are all ones which is reserved for both infinities and NaNs. Therefore we know that the count of normalized double-precision numbers is,

$$2 * (2^{11} - 2) * 2^{52}.$$

Problem 5.8: Consider the very limited system in which significands are only of the form,

$$1.b_1 b_2 b_3,$$

and the only exponents are 0,1, and -1. What is the machine precision ϵ for the system? Assuming that subnormal numbers are not used, what is the smallest positive number that can be represented in this system, what is the largest number that can be represented? Express you answers in decimal formatting.

Solution:
Calculating machine epsilon by definition,

$$\epsilon = 2^{-3} = .125.$$

Calculating the smallest positive number,

$$(1.000)_2 2^{-1} = .1000 = .5.$$

Calculating the largest positive number,

$$(1.111)_2 2^1 = 11.11 = 3.75.$$

Problem 5.9: Consider IEEE double-precision floating-point artithmetic, using round to nearest. Let a, b, and c be normalized double-precision floating-point numbers, and let $\oplus$, $\ominus$, $\otimes$, and $\oslash$ denote correctly rounded floating-point arithmetic.

1. is it necessarily true that $a \oplus b = b \oplus a$? explain why or give a counter example.

   Solution:
   Recall the definition of floating point arithmetic,

   $$a \oplus b = a + b + error.$$

   Therefore since regular addition is commutative we get,

   $$a \oplus b = a + b + error = b + a + error = b \oplus a.$$

2. Is it necessarily true that $(a \oplus b) \oplus c = a \oplus (b \oplus c)$? Explain or give a counterexample

   Solution:
   Consider the following Matlab output which uses IEEE double precision with round to nearest, **Console:**

   ```
   >> (10000000000000 + -10000000000000) + .00000001

   ans =

       1.0000e-08

   >> 10000000000000 + (-10000000000000 + .00000001)

   ans =

            0
   ```

   Obviously regular addition is associative, however IEEE addition is not because of the error defined in the definition.

3. Determine the maximum possible relative error in the computation,

   $$\frac{(a \otimes b)}{c},$$

Assuming that $c \neq 0$. Suppose $c = 0$ that are the possible values that the computation could be assigned.

Solution:
Recall the definition of relative error,

$$e_r = \frac{|x - round(x)|}{|x|}.$$

Recall that the maximum e for a round to nearest is,

$$e = |x - round(x)| \leq \frac{2^E \epsilon}{2}.$$

From the inequality,

$$2^E \leq |x|,$$
$$\frac{1}{|x|} \leq \frac{1}{2^E}.$$

Thus the maximum relative error is,

$$e_r = \frac{|x - round(x)|}{|x|} \leq \frac{\epsilon}{2}.$$

Since the computation,

$$\frac{(a \otimes b)}{c},$$

is a composition of 2 arithmetic operations we know that the maximum error is ϵ.

If we let $c = O$ then there are three cases, if $(a \otimes b)$ is positive then we return positive infinity. If $(a \otimes b)$ is negative then we return negative infinity. If $(a \otimes b)$ is zero we return NaN.

Problem 5.15: In the 1991 Gulf War, the Patriot missile defence system failed due to roundoff error. the troubles stemmed fromm a computer that performed the tracking calculations with an internal clock whose integer values in tenths of a second were converted to seconds by multiplying by a 24-bit binary approximation of one tenth.

1. Convert the binary number in (5.3) into a fraction call it x

Solution:
Converting the 24-bit binary approximation of one tenth,

$$x = \frac{209715}{2097152}$$

2. What is the absolute error in x,

   Solution:
   Subtracting,
   $$|\frac{209715}{2097152} - \frac{1}{10}| = \frac{1}{10485760}.$$

3. What is the time error in seconds after 100 hours of operation,

   Solution:

   $$|360000 - 3600000\frac{209715}{2097152}| \approx 3,240,000.$$

4. During the 1991 war, a Scud missile traveled at approximately Mach 5 (3750 mph). Find the distance that a Scud missile would travel during the time error computed in (c)

   Solution:

   $$\frac{3240000}{3600}3570 \approx 3,213,000.$$