1. (a) (10 points) Use the standard conversion algorithm (the description of which was given in class, and which can be found in any text on discrete math) to determine the binary representation of $(0.15)_{10}$. Note: this number has a repeating binary expansion.

Solution: Using the conversion algorithm given in class, the binary representation of $(0.15)_{10}$ is

$$0.00\overline{1001}$$

0.15
$\rightarrow$ 0.3
0.6
1.2
0.4
0.8
1.6
1.2
0.4
0.8
1.6
...

(b) (10 points) Show by explicit application of a geometric series (as done in class) that your answer in part (a) is correct.

Solution:

$$0.00\overline{1001}$$

$= \frac{1}{4}(0.1001 + 0.00001001 + ...)$
$= \frac{1}{4}(0.1001)(1 + \frac{1}{16} + \frac{1}{16} + \frac{1}{256} + ...)$
$= \frac{1}{4}(\frac{1}{2} + \frac{1}{16})(\sum\limits_{n=0}^{\infty}(\frac{1}{16}^{n}))$
$= \frac{1}{4}(\frac{1}{2} + \frac{1}{16})(\frac{1}{1-1/16}) = .15$

2. This question refers to the function `decimal_to_bin_str.m` distributed in class.

(a) (10 points) If the leading binary digit of the whole part of `a` is `0`, then there is a procedure performed in lines 68-78 of the script. If the leading binary digit is `1`, rather than `0`, then the script performs a different procedure (lines 84-92). What do each of these procedures do? How do these procedures differ? Why is it necessary to have two separate procedures for each of these cases?

Solution: The first bit of wb is 0 if the whole number part of `a` is 0, like -0.5 or 0.5. If the first bit is 0, the program shifts the decimal part of `a` over until 1 is the most significant digit so it can be represeneted in the mantissa. After that, a while loop converts the fractional part to binary using the conversion algorithm learned in class and makes it the mantissa.

The first bit of wb is 1 if the whole number part of `a` is not 0. The program takes everything from the second (index 2) digit on, and adds that to the mantissa. If there is room left in the mantissa, the program then converts the decimal part to binary using the conversion algorithm learned in class and appends it to the mantissa.

(b) (10 points) Suppose that n bits are used to store a floating point binary number with k digits of precision. Determine (with explicit calculation) the maximum value of the exponent, and (recalling that the bias is taken to be half of the maximum value of the exponent, rounded down) explain how line 96 correctly accounts for the bias.

Solution: The maximum value of any binary represented number with n bits is $2^n - 1$. In this case, the exponent has $n-k-1$ bits, because there are n bits, k bits are for the precision, and 1 bit is for the sign. Therefore, the maximum number that could be expressed with those bits is $2^{n-k-1} - 1$.

But with the floating point represention the exponent is the actual exponent minus the bias, with the bias being $\lfloor \frac{2^{n-k-1}-1}{2} \rfloor$. You could express the floor as $\lfloor 2^{n-k-2} - \frac{1}{2} \rfloor$.

Because 2^{n-k-2} will always be a whole number, subtracting $\frac{1}{2}$ and then taking the floor of that value is the same as $2^{n-k-2} - 1$, which proves that line 96 correctly accounts for the bias. Therefore, the actual maximum number the exponent could be would be $2^{n-k-1} - 1 - 2^{n-k-2} - 1$, or $2^{n-k-1} - 2^{n-k-2} - 2$.

3. This question refers to the function `bin_float_todec.m` distributed in class.

(a) (6 points) If one expects k digits of binary precision, how many digits of decimal precision should one expect? Explain.

Solution: For whole numbers: So if we have k bits of precision in binary then we can store 2^k values. We want to know how many digits of precision we will have in base 10, we will represent this unknown with x. We can write this as:

$$2^k = 10^x$$

Now we solve for x, we take the ceiling because we don't want to cut off any precision:

$$\lceil \log_{10}(2^k) \rceil = x$$

$$\lceil \log_{10}(2)k \rceil = x$$

For numbers with fractional parts: `K` digits of binary precision will yield you `K + 1` digits of decimal precision. An example (because we can't find a formula):

$1.1 (K = 1) \rightarrow 1.5$

$1.01 (K = 2) \rightarrow 1.25$

$1.001 (K = 3) \rightarrow 1.125$

(b) (6 points) Using your answer to part (a), explain why it may be necessary to include the command on line 14.

Solution: `VPA` in matlab does variable-precision arithmetic. The `digits` command sets the amount of precision needed - in our case that is `K + 1`, as shown in part (a).

(c) (8 points) Explain, in entirety, line 36.

Solution: Line 36 is performing the operation defined in any of the IEEE standards for converting binary float to decimal. That is, $(-1^{signum}) * (2 * exponent) * mantissa$. We use `vpa` to guarentee there is enough precision with no error during arithmetic, and to return the full precision number.

4. (a) (7 points) Convert 13.6 to a 9 bit binary floating point number with 5 digits of precision. Show your work.

Solution: A 9 bit binary floating point number with 5 points of precision will have 1 bit for the sign, 3 bits for the exponent, and 5 for the mantissa. 13.6 represented as a binary number is

$$1101.\overline{1001}$$

Normalized,

$$1.1011\overline{1001} \times 2^3$$

With three bits for the exponent, the bias is 3. Three was our exponent required for normalization, therefore the exponent part will be 6. The number is positive so the first bit will be 0. We then take 5 bits from binary point from the normalized representation, and get our binary floating point representation of 13.6 to be

$$011010110$$

(b) (7 points) Convert the floating point number obtained in part (a) back to decimal. Show your work.

Solution:

$$(-1)^0 \times 1.10110 \times 2^{exp-bias}$$
$$1.10110 \times 2^{6-3}$$
$$1.10110 \times 2^3 = 1011.10$$

$$13 + \frac{1}{2} = 13.5$$

(c) (6 points) Explain what happened.

Solution: The decimal part of 13.6 in binary is repeating, represented as $\overline{1001}$. When we specify only 5 bits of precision, we lose a lot of other bits needed to get close to 0.6. You'd need an infinite amount of bits to represent 0.6 perfectly in binary, as it is repeating.

5. The binomial coefficient
$$\binom{m}{k} = \frac{m!}{k!(m-k)!}$$
describes the number of ways of choosing a subset of k objects from a set of m elements.

(a) (6 points) Suppose decimal machine numbers are of the form

$$\pm 0.d_1d_2d_3d_4 \times 10^n, \qquad \text{with} \quad 1 \leq d_1 \leq 9,\ 0 \leq d_i \leq 9,\ \text{if } i = 2,3,4 \quad \text{and} \quad |n| \leq 15.$$

What is the largest value of m for which the binomial coefficient $\binom{m}{k}$ can be computed for all k by the definition without causing overflow?

Solution: Overflow is defined as a number having an exponent larger than the max value of n (15). The largest value of m for which the binomial coefficient $\binom{m}{k}$ can be computed for all k by the definition without causing overflow is 17. We know that $m!$ is the biggest term in the calculation, as $k \leq m$, $(m-k) \leq m$, and $\frac{m!}{k!(m-k)!} \leq m!$. Therefore, we can write a loop in MATLAB to find the highest value of m before $m!$ is greater than our max value, which is 0.9999×10^{15}.

Our code to do this was:

```
maxValue = 0.9999*(10^15);
m = 0;
while true
    if factorial(m) > maxValue
        m = m - 1;
        break
    end
    m = m + 1;
end
m
```

(b) (6 points) Show that $\binom{m}{k}$ can also be computed by

$$\binom{m}{k} = \left(\frac{m}{k}\right)\left(\frac{m-1}{k-1}\right)\cdots\left(\frac{m-k+1}{1}\right).$$

Solution: The binomial coefficient can be expanded to

$$\binom{m}{k} = \frac{1\times 2\times\cdots\times m}{(1\times 2\times\cdots\times k)(1\times 2\times\cdots\times(m-k))}.$$

This can be simplified by dividing the numerator by $(1\times 2\times\cdots\times(m-k))$, which results in

$$\binom{m}{k} = \frac{(m-k+1)\times\cdots\times m}{1\times 2\times\cdots\times k}.$$

You can split up all of the terms in this equation to be written as

$$\binom{m}{k} = \left(\frac{m}{k}\right)\left(\frac{m-1}{k-1}\right)\cdots\left(\frac{m-k+1}{1}\right),$$

resulting in the equation we want.

(c) (4 points) What is the largest value of m for which the binomial coefficient $\binom{m}{3}$ can be computed in part (b) without causing overflow?

Solution: We found that $m = 181707$, at which point any larger m would cause the product to overflow. The reason that we can get much larger values of m in this computation of the binomial coefficient is because we are cancelling out terms as we compute as opposed to computing whole factorials of m, which grow large extremely quickly, before cancelling anything out.

We used similar code to part (a):

```
maxValue = 0.9999*(10^15);
m = 0;
while true
    product = 1;
    for i=0:2
        product = product * ((m-i)/(3-i));
    end
    if product > maxValue
        m = m - 1;
        break
    end
    m = m + 1;
end
m
```

(d) (4 points) Use the equation in (b) and four-digit chopping arithmetic to compute the number of possible 5-card hands in a 52-card deck. Compute the actual and relative errors.

Solution: We want to compute $\binom{52}{5}$ using the equation in (b). This would give us

$$\binom{52}{5} = \frac{52}{5} \times \frac{51}{4} \times \frac{50}{3} \times \frac{49}{4} \times \frac{48}{1}.$$

We then compute each of the values inside of the parentheses, using four-digit chopping:

$$\frac{52}{5} = 10.4$$
$$\frac{51}{4} = 12.75$$
$$\frac{50}{3} = 16.66$$
$$\frac{49}{2} = 24.5$$
$$48 = 48$$

We then multiply each of those together, still with four-digit chopping:

$$10.4 \times 12.75 = 132.6$$

$$16.66 \times 24.5 = 408.1$$

$$132.6 \times 408.1 = 54110$$

With our final answer being:

$$54110 \times 48 = 2597000$$

The actual value of $\binom{52}{5}$ is 2598960. Therefore, our actual error is

$$|2598960 - 2597000| = 1960$$

and our relative error is

$$\frac{|2598960 - 2597000|}{2598960} = 7.5415 \times 10^{-4}.$$