Problem 1.

Problem 2.

```
/*
Alleged solution for assignment, this version takes in the
number as a double but any number format would be fine as long as you are
consistent.
*/

#include<stdio.h>

int main(void)
{
    double value; // Value to be read in and worked with

    printf("Please enter a number: ");
    scanf("%lf", &value);

    // Number games
    printf("%lf times 2 is: %lf\n", value, value * 2);
    printf("%lf + 7 = %lf\n", value, value + 7);
    printf("2(%lf)^3 + 14 - 2(%lf) = %lf\n",
      value, value, 2*(value * value * value) + 14 - 2*(value));
    // and more if interested ...
    return 0;
}
```

Problem 3.

```
#include<stdio.h>

int main(void)
{
    int n, m; // The two numbers to be inputted
    printf("Please enter a number: ");
    scanf("%d", &n); // get the first number
    printf("Please enter another number: ");
    scanf("%d", &m); // get the second number
    // Calculate q
    int q = n / m;
    // Calculate r
    int r = n % m;
    // Do the final printing
    printf("%d = %d * %d + %d\n", n, q, m, r);

    /*
      Could also have done all calculations and printing in one line, like so
      printf("%d = %d * %d + %d", n, n / m, m, n % m);
     */
    return 0;
}
```

Problem 4. Here's a truth table argument for the first law.

P	Q	$P \vee Q$	$\neg(P \vee Q)$	$\neg P$	$\neg Q$	$\neg P \wedge \neg Q$
F	F	F	**T**	T	T	**T**
F	T	T	**F**	T	F	**F**
T	F	T	**F**	F	T	**F**
T	T	T	**F**	F	F	**F**

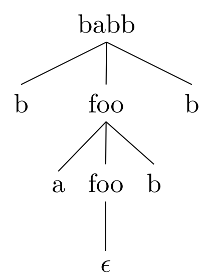
Since the output of both expressions is the same for all possible assignment of truth values to P and Q, $\neg(P \vee Q) \iff \neg P \wedge \neg Q$.

Problem 5.

$$\begin{aligned}
f(20) &= 20 \cdot f(10) && \text{Since } 20 \mod 2 = 0 \\
&= 20 \cdot 10 \cdot f(5) && \text{Since } 10 \mod 2 = 0 \\
&= 20 \cdot 10(5 + f(4)) && \text{Since } 5 \mod 2 = 1 \\
&= 20 \cdot 10(5 + 4 \cdot f(2)) && \text{Since } 4 \mod 2 = 0 \\
&= 20 \cdot 10(5 + 4 \cdot 2 \cdot f(1)) && \text{Since } 2 \mod 2 = 0 \\
&= 20 \cdot 10(5 + 4 \cdot 2 \cdot 1) && \text{Since } 1 = 1 \\
&= 20 \cdot 10(5 + 8) \\
&= 200(13) \\
&= 260
\end{aligned}$$

Problem 6. (a) Note that the foo definition seems to deal with the start end end characters of a given string. aba is not the empty string so we will check the other clauses. Since the starting and ending characters of aba are a, and a, we will proceed with the clause $a \circ foo \circ a$. From here we must examine the substring left after removing the starting and ending characters from aba, which is b. b does not fit any of the clauses in the foo definition, so aba cannot be a foo.

(b) Proceeding as above, $babb$ is clearly not the empty string, but it seems to fit the clause $b \circ foo \circ b$. Looking at the foo definition again, we can further break this down into $b \circ a \circ foo \circ b \circ b$. We can now use the empty string clause leaving us with $b \circ a \circ \epsilon \circ b \circ b = babb$. So $babb$ is a foo. This process can be organized using a tree.



(c) Every case in our foo definition involves either the empty string or a string made of some combination of the characters a and b and another foo. The key to describing what a foo is is to realize that anything that is a foo will successfully go through the process outlined in part (b) and so will end up being some number of as and bs concatenated together. So we should look at the as and bs in the cases of our foo definition and try to notice a pattern. In each case, there are always two literal characters or the empty string, so the length of any foo will be even (we can also see this by examining the process in part (b) which pairs off as and bs). The order of the as and bs does not really matter since the cases cover every size two permuation of a and b (e.g. aa, ab, ba, bb). So a foo is any even length string of as and bs.

Remark 1. In the field of *Formal Language Theory*, the set of all foos could be considered a *language* with the *alphabet* $\{a, b\}$. I defined a foo in a way that is similar to a *context-free grammar*, which is what the foo language is. Context-free grammars are very useful throughout computer science, mathematics, and linguistics for their ability to mathematically describe the behavior of certain languages.