Attempt any **three** of the **six** problems below. The homework is graded on a scale of 100 points, even though you can attempt fewer or more points than that. Your recorded grade will be the total score on the problems that you attempt.

1. (**20 points**, Sipser problems 5.17 and 5.18)
   (a) (**10 points**) Prove that the Post Correspondence Problem is decidable if the alphabet is unary, i.e., $\Sigma = \{\mathtt{1}\}$.
   (b) (**10 points**) Prove that the Post Correspondence Problem is undecidable if the alphabet is binary, i.e., $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

2. (**30 points**, Sipser problem 5.21) Let $AMBIG_{CFG} = \{G \mid G$ describes an ambiguous CFG$\}$. Show that $AMBIG_{CFG}$ is undecidable. (Hint: Use a reduction from PCP. Given a PCP instance

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots, \left[\frac{t_k}{b_k}\right] \right\},$$

construct a CFG G with the rules

$$\begin{array}{lcl} S & \rightarrow & T \mid B \\ T & \rightarrow & t_1 T \mathtt{a}_1 \mid \ldots \mid t_k T \mathtt{a}_k \mid t_1 \mathtt{a}_1 \mid \ldots \mid t_k \mathtt{a}_k \\ B & \rightarrow & b_1 B \mathtt{a}_1 \mid \ldots \mid b_k B \mathtt{a}_k \mid b_1 \mathtt{a}_1 \mid \ldots \mid b_k \mathtt{a}_k, \end{array}$$

where $\mathtt{a}_1 \ldots \mathtt{a}_k$ are new terminal symbols. Prove that this reduction works.)

3. (**35 points**, Sipser problem 5.26) Define a *two-headed finite automaton* (2HDFA) to be a deterministic finite automaton that has two read-only, bidirectional heads that start at the left-hand end of the input tape and can be independently controlled to move in either direction. The tape of a 2HDFA is finite and is just large enough to contain the input plus a left-endmarker, $\vdash$, and a right-endmarker, $\dashv$. A 2HDFA may not move either of its heads beyond either delimeter. A 2HDFA accepts by entering a special accept state.
   (a) **(10 points)** Describe a 2HDFA that recognizes the language $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$. You don't need to specify all the details of the transition function. Just write a few sentences explaining how it works.
   (b) **(10 points)** Let $A_{2HDFA} = \{M\#w \mid M$ describes a 2HDFA that accepts $w\}$. Show that A_{2HDFA} is Turing-decidable.
   (c) **(15 points)** Let $E_{2HDFA} = \{M \mid M$ describes a 2HDFA such that $L(M) = \emptyset\}$. Show that E_{2HDFA} is not Turing-decidable. (Hint: use computational histories.)

4. (**35 points**, See Sipser problem 5.31) Let

$$f(x) = \begin{cases} 3x+1, & \text{if } x \text{ is odd} \\ x/2, & \text{if } x \text{ is even} \end{cases}$$

for any integer $x \geq 0$. Starting from x, obtain the sequence x, $f(x)$, $f(f(x))$, $\ldots$. Stop if you ever reach 1. This sequence is known as the "hailstone" sequence for x. For example if $x = 23$, then you get the sequence: 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1. Extensive comptuer tests have showen tha every starting point from 1 through 2.88×10^{18} produces a seqence that ends in 1 (see http://en.wikipedia.org/wiki/Collatz_conjecture). The Collatz conjecture is that all positive starting points end up at 1, and this conjecture is unsolved.

Show that the Collatz conjecture is Turing-reducible to $TOTAL_{TM}$.

Note: Sipser seems to be asking to show a reduction to A_{TM} but I couldn't think of any way to do that.

5. (**35 points**, Sipser problem 5.34) Let P be a PDA and $WW = \{ww \mid w \in \{\mathtt{0},\mathtt{1}\}^*\}$. Use computational histories to show that the question of whether P accepts some string in WW is undecidable.

6. (**45 points**) The RSA encryption method relies on the assumption that it is difficult to factor large numbers. However, no one knows whether or not there is a polynomial time algorithm for factoring. However, there does exist a polynomial time algorithm that will determine whether or not the integer represented by N (a binary string) is prime or composite, but if N is composite, this algorithm doesn't determine the factors.

   Now, consider a TM, M, that does the following when run with input N, a binary encoding of an integer:

   1. Check if N is prime (e.g., using the algorithm described above). If N is prime, M writes the string "`0`" on its tape and halts. Otherwise, it continues to step 2.
   2. Find f a factor of N with $1 < f < N$. M then writes the binary encoding of f on its tape and halts.

   Describe how you can implement the second step of this algorithm with a method that will find f in polynomial time iff there is a polynomial time algorithm for factoring. In other words, your algorithm should be one that runs in polynomial time if factoring is polynomial, and in super-polynomial time otherwise.

   Hints: (1) Use diagonalization. (2) Remember that big-O analysis ignores constant factors, even ***really big*** ones.