Problem 1

Write a program that outputs its own source code:

;;;The SELF program!!!
(define self
  (lambda (w)
    ((lambda (w) (list (quote lambda)
      (quote (w))
      (list w ((lambda (w) (list (quote quote) w)) w)))))
    (quote
      (lambda (w) (list (quote lambda)
        (quote (w))
        (list w ((lambda (w) (list (quote quote) w)) w))))))))

;;;Explanation...

;;;PRINT_a  <=>  f(w) = a
;;;<PRINT_a>  <=>  (quote a)

;;;TM_q  <=>  q(w) = <PRINT_w>
;;;<TM_q>  <=>  (lambda (w) (list (quote quote) w))

;;;TM_p  <=>  w(q(w))
;;;<TM_p>  <=>  (lambda (w) (list (quote lambda)
;;;               (quote (w))
;;;               (list w (<TM_q> w))))

;;;SELF  <=>  TM_p(PRINT_<TM_p>(w))
;;;<SELF>  <=>  (lambda (w) (<TM_p> (quote <TM_p>)))
Problem 6.6

In the fixed-point version of the recursion theorem, let the transformation $t$ be a function that interchanges the states $q_{\text{accept}}$ and $q_{\text{reject}}$ in Turing machine descriptions. Give an example of a fixed point for $t$.

- Let the fixed point machine be $\langle M \rangle$. If $M$ halts on any input $w$, then clearly it cannot be a fixed point, since the language $M$ recognizes and the language $t(\langle M \rangle)$ recognizes must differ in $w$.

- However, suppose $M$ loops infinitely on all inputs. Then $t(\langle M \rangle)$ must also loop infinitely, because it is the same machine but with accept and reject states swapped. Therefore $M$ is a fixed-point. So any machine which loops infinitely on all inputs (for example, a machine which can never leave the start state) is a fixed-point.

Busy Beaver Problem

If a given Turing machine halts when started on a blank tape, then define the productivity of the Turing machine as the length of the string it leaves on the tape. If the machine does not halt, define its productivity as zero. Let $p(n)$ be the productivity of the most productive $n$-state Turing machine. Show that this function is uncomputable.

- Suppose the productivity function $p(n)$ is computable. Then we can derive a contradiction by constructing a machine $B$.

  $B =$ "On blank input:
  1. Obtain own description $\langle B \rangle$ via the recursion theorem.
  2. Count number of states $n$ of $\langle B \rangle$
  3. Compute $p(n)$
  4. Add 1 to get $p(n)+1$
  5. Print $p(n)+1$ 1s on the end of the tape.

- Now $B$ is an $n$-state machine that produces more than the productivity function says it should be able to, so we have a contradiction.