Due: Friday, March 14, beginning of tutorial

NOTE: Each problem set counts 15% of your mark, and it is important to do your own work. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. Copying assignments is a serious academic offence and will be dealt with accordingly.

1. Show that if \( f(x) = \mu y[g(x, y) = 0] \) and \( g \) is a computable function, then \( f \) is a computable function. Do this by giving an RM (Register Machine) program for computing \( f \), using an RM program \( P_g \) which computes \( g \). (You may use the copy macro \( R_i \leftarrow R_j \).)
   (This is essentially Exercise 7 page 61 of the Notes.)
   The idea is to implement the pseudo-code given at the bottom of page 60 of the Notes (where \( \vec{x} = x \), i.e. \( x \) is a single variable). Be sure to respect our input/output conventions for computing functions on register machines.

2. Given a function \( f(x) \) define a function \( \text{Max}_f(x) = \max\{f(0), f(1), \ldots, f(x)\} \). Show that if \( f \) is primitive recursive then \( \text{Max}_f \) is primitive recursive. In addition to the definition of primitive recursive, you may use results in the Notes “Computability Theory”.

3. Do Exercise 14, page 70 of the Notes: Use a diagonal argument to prove that the function
   \[ H(x) = \mu yT(x, x, y) \]
   has no total computable extension.
   (The proof is similar to the proof of the Theorem on page 69. You may use the idea in this proof, but do not use the theorem itself.)

4. We say that a function \( f : \mathbb{N} \rightarrow \mathbb{N} \) is nondecreasing if
   \[ x \leq y \Rightarrow f(x) \leq f(y), \text{ for all } x, y \in \mathbb{N} \]
   Prove that a set \( A \subseteq \mathbb{N} \) is recursive iff \( A = \emptyset \) or \( A \) is the range of some total computable unary nondecreasing function \( f \). (Recursive sets are defined on page 71 of the Notes.) Give a careful proof, without using Church’s thesis. (You may use the fact that the class of total computable functions is closed under primitive recursion.) **Hint:** It may help to consider separately the case in which \( A \) is finite.