Assignment 1

Complete the following problems and turn in a hardcopy of your solutions by the beginng of class on January 21$^{\text{st}}$ (5:00 pm).

1. **4 points.** Consider an extension of diML that provides support for references, arrays, and while loops (each defined as we did in class). Call this language diml-RAW. We wish to extend diml-RAW so that it supports `for` and `foreach` loops. The `for` loops should have the form `for(e1; e2; e3) { e4 }`, and should behave as `for` loops in C, except they should evaluate to `false` when the loop terminates (like `while` loops in diml-RAW). The `foreach` loops should have the form `foreach(x in e1) { e2 }` and operate as shown below (note that `foreach` loops should also evalaute to false when the loop terminates). Provide the static and dynamic semantics of both `for` and `foreach` loops, keeping consistent with the existing language semantics of diml-RAW.

   The following function should, when given an integer array, add 1 to every element in the array.

   ```
   fun f (xs : int array) : bool = foreach (x in xs) {x := !x + 1;}
   ```

2. **6 points.** Let's assume that you have already defined a single-step judgement (i.e., $e \rightarrow e'$) for some language L. Now, define the multi-step judgement $e \rightarrow^* e'$. Next, prove the following theorem, which states that the alpha-equivalence relationship is preserved by the multi-step relation.

   α-safety: $((e_1 =_\alpha e_2) \wedge (e_1 \rightarrow^* e_1')) \Rightarrow \exists e_2' : (e_2 \rightarrow^* e_2' \wedge e_1' =_\alpha e_2')$

   In your proof of the alpha-safety theorem, you may assume that the following two lemmas hold:

   α-progress: $((e_1 =_\alpha e_2) \wedge (e_1 \rightarrow e_1')) \Rightarrow \exists e_2' : e_2 \rightarrow e_2'$

   α-preservation: $((e_1 =_\alpha e_2) \wedge (e_1 \rightarrow e_1') \wedge (e_2 \rightarrow e_2')) \Rightarrow e_1' =_\alpha e_2'$

3. **1.5 points extra credit.** Prove that the α-progress and α-preservation lemmas hold for the call-by-name untyped lambda calculus.