Assignment #4

Name: ______________________ ID: ______________________

This assignment has **4** questions, for a total of **32** marks.

Recall the following acronyms: SOS (structural operational semantics), COS (contextual operational semantics), SM (small step), BG (big step), CBV (call by value), CBN (call by name).

Question 1: **Progress** .... 8 marks

Write the proof for the progress theorem for the cases related to locations [2], allocation [2], dereferencing [2] and update [2].

Question 2: **Program equivalence** .... 10 marks

For each of these programs, tell if they are equivalent or not. If they are equivalent, show what they reduce to, no matter the input. If they are not, argue why and if possible, show a context that tells them apart.

1. $z : \mathit{Ref}\ (N \to N)$ [2]
   $t_1 = \lambda x : N.\, !z\ 0; 2 + x$
   $t_2 = \lambda x : N.\, \mathit{if}\ x > 0\ \mathit{then}\ x + 2\ \mathit{else}\ !z\ x; x + 2.$
2. $t_1 = \mathit{let}\ x = \lambda y : \forall\alpha.\alpha \to \alpha.\, \lambda z : N.\, y\ [N]\ (z + 1)\ \mathit{in}\ x$ [2]
   $t_2 = \lambda y : \forall\alpha.\alpha \to \alpha.\, \lambda x : N.\, (y\ [N]\ x) + 1.$
3. $f : (\mathit{Ref}\ N) \to N$ [2]
   $t_1 = \mathit{let}\ x = \mathit{new}\ 0\ \mathit{in}\ f\ x; !x$
   $t_2 = \mathit{let}\ x = \mathit{new}\ 1\ \mathit{in}\ f\ (\mathit{new}\ 0); x := (!x - 1).$
4. $r : \mathit{Ref}\ N$ [2]
   $t_1 = \mathit{let}\ x = !r\ \mathit{in}\ \mathit{let}\ y = \mathit{new}\ x\ \mathit{in}\ r := !y; !y$
   $t_2 = \mathit{let}\ x = \mathit{new}\ 0\ \mathit{in}\ \mathit{let}\ y = !x; !r\ \mathit{in}\ y.$
5. $t_1 = \lambda x : N.\, \langle x, 1\rangle .1$ [2]
   $t_2 = \mathit{let}\ x = \Lambda\alpha.\lambda x : \alpha.\, x\ \mathit{in}\ x.$

Question 3: **A Private Memory for ASM** .... 6 marks

Add a private memory to ASM. The domain of the memory becomes integers, so positive and negative numbers. Negative integers represent a private memory. [2]

Modify the semantics of ASM to reflect the following access control policy:

- if the program counter is within the address range 0 to 100, then any read or write to the private memory succeeds, otherwise any read or write returns 0. [4]

Question 4: **Labelled ASM Functions** .... 8 marks

Add named functions to ASM. ASM programs become a list mapping names to codebases. [2]

ASM instructions now include

- calling a function whose name is statically known; [2]
- calling a function by jumping to the address where it starts (the address value is read from a register); [2]
- returning from a called function. [2]

The semantics must assign an address to each function name, so that it can resolve the meaning of calling a function, i.e., jumping to the address where it lays.