# CMPT 383 Comparative Programming Languages

## Homework 6

This homework is due by 11:59pm PT on Tuesday Mar 18, 2025. No late submission is accepted. Please save your answers in a single file called `H6_SFUID.pdf` and submit it to Canvas. You may also write on paper and scan it (or take a picture) into a PDF. Please make sure the text is readable.

Requirements:

- Please include an environment in the judgments even if it is not used.

1. (20 points) Consider the FUN language that we learned, provide a big-step operational semantics to expression $e_1$`<=`$e_2$. The expression evaluates to `true` if the value of $e_1$ is less than or equal to the value of $e_2$. Otherwise, it evaluates to `false`.

2. (20 points) Consider the FUN language, prove the following expression evaluates to 3 with respect to the big-step operational semantics that can handle recursion.

$$\texttt{let}\ x = 2\ \texttt{in}\ (1 + x)$$

Note that the parentheses are just used to show the precedence. You do not need to show the steps for parentheses in the proof.

3. (30 points) Suppose we add a program construct called `testSign` to the FUN language with the following syntax

$$\begin{array}{rcl} e & ::= & \ldots \quad \text{(all existing productions in FUN)} \\ & | & \texttt{'testSign'}\ e\ e\ e\ e \end{array}$$

The evaluation result of `testSign` $e_1$ $e_2$ $e_3$ $e_4$ is

- the result of $e_2$, if $e_1$ evaluates to a negative number
- the result of $e_3$, if $e_1$ evaluates to zero
- the result of $e_4$, if $e_1$ evaluates to a positive number

Provide a big-step operational semantics for `testSign`.

4. (30 points) Consider the `testSign` in Quesiton 3, provide a small-step operational semantics for `testSign`. Note that for expression `testSign` $e_1$ $e_2$ $e_3$ $e_4$, the expression $e_1$ should be evaluated first. You can assume the small-step operational semantics for other FUN constructs already exists.