# Assignment 5: Type Systems

**CS - 456 – Programming Languages**

November 3, 2014

## General Information

- Due Date: 11/17/2014 by midnight.
- Exercises 1, 2 and 4 are individual. *Exercise 3 can be submitted by pairs.*
- Use Piazza for Q&A.

## Submission

- Add each of the files required below in a single directory with the following naming structure: {lastname1}-{lastname2}-hw5.
  Submit a single file with the directory: {lastname1}-{lastname2}-hw5.zip (or .tgz)
- Exercise 3 can be solved in pairs. Please write a note in your submission stating if you have worked with a partner. Also add a note in your BlackBoard submission.
- **Important:** Respect the naming conventions. If you change the names of the functions, they will not be evaluated by the grader scripts.

## Programming Exercises

1. Typing arrays in Typed Impcore. Do exercise 1 of Chapter 6 of Ramsey's book (pg. 277). (Hint: your solution should be rather small.)
   **Turn in:** Add your solution in a file named `timpcore.sml`.
2. Typed $\mu$Scheme: Solve exercise 11 of Chapter 6 of Ramsey's book (pg. 279).
   **Turn in:** Add your solution in a file named `11.scm`.
3. Typed $\mu$Scheme: Solve exercise 13 of Chapter 6 of Ramsey's book (pg. 279). This exercise takes considerable more amount of time and effort than the other exercises. Below you will find some advice on how to approach this exercise.
   **Turn in:** Add your solution in a file named `tuscheme.sml`.

**Advice:**

- Testing: You may want to comment out the initial basis when writing your type checker. You can do that by replacing the line

  ```
  val basis = (* ML representation of the initial basis *)
  ```

  with

  ```
  val basis_included = false
  val basis = if not basis_included then [] else
  ```

  This will not include the initial basis when testing. When you finish with the implementation of the type-checker reset `basis_included`. Make sure that the initial basis is included in your final submission.
- Study Subsection 6.6.4. of Ramsey's book.
- Write the typechecker piecemeal.
- Start by writing a all the cases of the syntax, raising `LeftAsExercise` exception in each of them initially. You can then write one case at a time, and test them individually. Below is an order advisable to tackle the implementation.
- Implement the checking of literal numbers and booleans.
- Add conditional expressions.
- Implement VAL and EXP. This will allow you to test some simple conditional expressions.
- Implement the checking of function application. You will be able to test arithmetic and comparison expressions.
- Check LET and VAR.
- Similarly to LET you can implement LAMBDA. To create function types use the function `funtype`.
- Implement SET, WHILE and BEGIN.
- Type LETSTAR which can be defined as syntactic sugar of LET.
- Type DEFINE and VALREC (remember that define is syntactic sugar of valrec).
- Type TYAPPLY and TYLAMBDA.

## Theory Exercise

4. Solve exercise 5 of Chapter 6 of Ramsey's book (pg. 278). Pay attention to the typing of the nil list.

   **Turn in:** Add your solution in a file named `lists.pdf`.

## General Evaluation Criteria

The evaluation criteria is similar to previous assignments. Write clear code, you can use the following style guide for instance `http://www.cs.cornell.edu/courses/cs312/2005sp/handouts/style.htm`. Test your own implementation. Test cases are not provided. You can provide your test cases in your submission.