

Project 3: Scheme Interpreter Extension

CSC 4101, Fall 2009

Due: 5 December 2009

For this project, you will extend the functionality of the Scheme interpreter from Project 2 by writing additional built-in function in Scheme.

Implement at least the following Scheme functions:

- the comparison operations `eqv?` and `equal?`;
- the n-ary integer comparison operations `=`, `<`, `>`, `<=`, `>=`;
- the test predicates `zero?`, `positive?`, `negative?`, `odd?`, `even?`;
- the n-ary arithmetic operations `max`, `min`, `+`, `-`, `*`;
- the boolean operations `not`, `and`, `or`;
- the list functions `caar`, `cadr`, ... `cddddr`, `list`, `length`, `append`, `reverse`;
- the set and association list operations `memq`, `memv`, `member`, `assq`, `assv`, `assoc`;
- the higher-order functions `map` and `for-each`.

Extend the main function of Project 2. After populating the built-in environment with all the `BuiltIn` objects, it would read the Scheme definitions from a file `ini.scm` into the built-in environment. After reading the initialization file, your interpreter would create the top-level environment and start reading user input.

If your Project 2 is not working, you can develop the above Scheme code in Scheme48. To do so, you will first need to define the built-in functions from Project 2. E.g., you first need to define

```
(define builtin-+ +)
(define (b+ x y) (builtin-+ x y))
```

in an environment in which `+` is the built-in Scheme48 operation. Then you can define your own version of `+` using only `b+`.

Administrative Stuff

Program this project individually, not in groups. Put your code into a file `ini.scm`, but do not include the definitions for the built-ins from Project 2 (such as `b+`, etc.). Submit this file together with a `README` file with anything you want to tell the grader.