CPSC 3740—Fall 2024
Course Project

Due: December 6, 2024 11:55pm

1 Description

In this group project, you will implement an interpreter for a subset of Racket inside Racket. The main goal of this project is the implementation of the function `startEval`, which takes one argument being a Racket program. The function evaluates the given argument and returns the result. You can assume that the given Racket program is syntactically correct.

2 Language Description

Your interpreter `startEval` needs to handle the following constructs of Racket:

Constants and variables: numbers, variables (e.g. `x`), and `quote` (e.g. `(quote a)`)

Arithmetic operators: `+`, `-`, `*`, `/`. Only binary operators on numbers are supported (e.g. `(+ 1 2)`)

Relational operators: `equal?`, `=`, `<=`, `<`, `>=`, `>`. Except for `equal?`, only binary operators on numbers are supported.

Lists: `car`, `cdr`, `cons`, `pair?`

Conditional: `if`

Lambda expression: `lambda`. Only implement the plain lambda syntax without support for keyword or optional arguments. Assume there is only one expression in the body.

Function application: applying a lambda expression to arguments

Local binding: `let`, `letrec`. Assume there is only one expression in the body.

3 Written Report

You will also submit a written report (at most 5 pages) addressing the following elements:

- How did you organize your program?
- What are the key data structures you used and what were they used for?
- Are there any limitations/bugs?
- How did you test your interpreter? Include 5 sample test runs illustrating the different aspects of functionalities of the interpreter. Do not simply use the supplied test case(s) below.

4 Grading

Your interpreter will be graded on functionality and readability. It is expected that you make use of functions to organize your code, and that you document each function and its parameters. You are also expected to document the main idea of any non-trivial algorithm or data structures involved.

Functionality: 65%

Constants and variables: 5%

Arithmetic operators: 5%

Relational operators: 5%

Lists: 5%

Conditional: 5%

Lambda expression: 10%

Function application: 10%

Local binding: 20%

Documentation: 15%

Report: 20%

5 Submission

Your interpreter should be submitted on Moodle (instructions will be posted). Your interpreter should be stored in a file named `startEval.rkt`.

6 Group Membership

You may choose to complete this project in groups of up to 4 people. **Please indicate to the instructor by November 1, 2024 the group composition.** All members of the group will receive the same grade. It is your responsibility to ensure each member contribute to the project. Note that the grading scheme is the same regardless of the size of the group.

7 Example

In Racket, typing the following:

```
(print
 (startEval
 '(letrec ((fact
            (lambda (x)
              (if (= x 0) (quote 1)
                (* x (fact (- x 1)))))))
              (fact 10))))
```

should print `3628800`.

8 Hints

This is a very large project. You may want to start by implementing one functionality at a time and ensuring that the functionality is correct before continuing.

However, before you begin coding you need to think about how you would maintain the list of current local bindings (e.g. `let` and `letrec`) and apply them as needed. You may start the evaluation with an empty list of local bindings.

9 Remarks

- **DO NOT** use `eval` in Racket—you must implement your own evaluation funciton.
- **DO NOT** use any Racket functions that can cause side effects such as setting variable values. You may of course use `define` to define functions.