CSE 401: Section 2

Starter-kit

If you have not already done so, please download and extract the code at the following link. Use your CSE email to access it.

http://tinyurl.com/sec2-starter-kit

The starter kit provided implements a compiler for a very simple language. You can run the compiler using nodejs via command "$ node main.js inputFile" or "$ nodejs main.js inputFile". If necessary, detailed instructions are available on the piazza post and in the email sent to you.

Note: The lexical parser used by the compiler is remotely deployed, please make sure you are connected to the internet.

Grammar for our language

```
P       ::=     S*                          /* List of statements */

S       ::=     E                           /* Expression */
        |       Id = E                      /* Variable assignment */
        |       def Id = E                  /* Var. Declaration */
        |       def Id ( Id* ) { S* }       /* Function Definition */
        |       print E                     /* Print E to console */
        |       error E                     /* Print E and exit */
        |       if ( E ) { S* } else { S* } /* If-else statement */
        |       while ( E ) { S* }          /* While loop */
        |       for ( Id in E ) { S* }      /* For loop */

E       ::=     Num                             /* A number literal */
        |       String                      /* A string literal */
        |       Id                          /* A variable */
        |       E + E                       /* Addition */
        |       E - E                       /* Subtraction */
        |       E < E                       /* LT comparison */
        |       ite(E, E, E)                /* Ite Conditional */
        |       lambda ( Id* ) { S* }       /* Anonymous function */
        |       E ( E* )                    /* Function call */

Num     ::=     /-?[0-9]+/                  /* A Number */
Id      ::=     /[a-zA-Z_][a-zA-Z_0-9]*/    /* An Identifier */
String  ::=     /"[^"]*"|'[^']*'/           /* A String */
```

Exercise 1: Extending the core language (<5 minutes)

The interpreter currently only supports arithmetic addition. Extend the interpreter in `interpreter.js` to support subtraction (“-”) and less than comparison (“<”).

Remember: There is no special type for Booleans. As in C, 0 is False and all non-zero numbers are True. Null is also interpreted as false.

Exercise 2: De-sugaring (5 minutes)

If, while-loop and for-loop constructs are not part of the core language and thus need to be decomposed into the core language before the AST is handed to the interpreter. In order to support new constructs, you need to provide re-write rules in our de-sugaring DSL. Code responsible for using these rules to transform your AST has already been provided.

De-sugaring DSL

You are required to write re-write rules or expansion expressions using the our desugaring DSL. The expansion expressions look exactly like a normal expressions, except for the use of metavariables. These metavariables come in two forms:

- New identifiers: %u1, %u2, etc…
- AST subtrees %name, %iterable, %body. So for an AST node { “condition”: “xyz”, “name”: , “abc” }, “%condition” in your expression will be replace by the value “xyz”.

You can explore the structure of the ASTs by printing them to console. As an example, the if construct has already been done for you.

Exercise 3: Scoping (15 minutes)

As a final exercise, we will be implementing variable definition, variable lookup and variable modification. Variables should have static scoping i.e. the definition from the most local environment must be used.

Step 1: Add local variable declaration and assignment support by completing the envBind(), envUpdate() and envLookup() functions in `environment.js`.
Step 2: Fix the "call" case in evalExpression(). A new frame must created for each call. You will also need to implement envExtend(). Hint: Pay attention to handling parameters.
Step 3: Implement closures support for lexical scoping.

To print a runtime error, use the snippet ‘throw new ExecError(“error_message”);’.