CSE 6341: LISP Interpreter Project, Part 4

Overview

Project 4 does not involve any execution of LISP code, but rather a very limited form of static type checking. You only need to use the scanner and parser from your previous projects; no evaluation will be needed. As before, *your submission should compile and run in the standard environment on stdlinux.*

Input

We will consider a very limited subset of possible LISP behavior. We will allow values of three types: Nat, List(Nat), and Bool. There are no user-defined functions. The only built-in constructs we will use are T, NIL, CAR, CDR, CONS, ATOM, EQ, NULL, INT, PLUS, LESS, and COND. In addition, we will use the literal atom F to represent Bool value "false". NIL will be used only to represent an empty list of integers.

Typing relation

The type checking should be done based on the following set of inference rules for the typing relation.

Axioms

const : Nat	T : Bool	F : Bool	NIL : List(Nat)

Note: "const" denotes a numeric atom

Other Inference Rules

$$\frac{s : \text{List(Nat)}}{(\text{CAR } s) : \text{Nat}} \qquad \frac{s : \text{List(Nat)}}{(\text{CDR } s) : \text{List(Nat)}} \qquad \frac{s_1 : \text{Nat} \quad s_2 : \text{List(Nat)}}{(\text{CONS } s_1\ s_2) : \text{List (Nat)}}$$

$$\frac{s : T}{(\text{ATOM } s) : \text{Bool}} \qquad \frac{s_1 : \text{Nat} \quad s_2 : \text{Nat}}{(\text{EQ } s_1\ s_2) : \text{Bool}} \qquad \frac{s : \text{List(Nat)}}{(\text{NULL } s) : \text{Bool}}$$

$$\frac{s : T}{(\text{INT } s) : \text{Bool}} \qquad \frac{s_1 : \text{Nat} \quad s_2 : \text{Nat}}{(\text{PLUS } s_1\ s_2) : \text{Nat}} \qquad \frac{s_1 : \text{Nat} \quad s_2 : \text{Nat}}{(\text{LESS } s_1\ s_2) : \text{Bool}}$$

$$\frac{s_1 : \text{Bool} \quad s_2 : T \quad s_3 : \text{Bool} \quad s_4 : T \quad \ldots \quad s_{2k-1} : \text{Bool} \quad s_{2k} : T}{(\text{COND } (s_1\ s_2)\ (s_3\ s_4)\ \ldots\ (s_{2k-1}\ s_{2k})) : T}$$

In these rules, T denotes any of the three types Nat, Bool, and List(Nat). In the last rule, all occurrences of T refer to the same type – that is, there is one instance of the rule in which all T are replaced with Nat, another instance of the rule in which all T are replaced with Bool, etc.

These static typing rules are more restrictive than the dynamic typing rules you implemented in Project 2. For example, expressions such as (EQ 5 T), (NULL 5), (COND ((CONS 4 NIL) T) (T 5)) will evaluate successfully in your Project 2 implementation, but will not be well-typed according to these typing rules, and will be rejected by the typechecker.

Type checking

You need to implement a typechecker for this language. Any well-typed expression should be accepted and printed using the printing approach from previous projects. Do not evaluate the expression, just print it. If the expression is not well-typed, print an error message as in Projects 1-3, and exit to the OS.

Testing Your Project

You need to prepare and submit two files, ValidTests and InvalidTests. Each invalid test should result in a type checking error.

Project Submission

On or before 11:59 pm, **November 24 (Tuesday),** you should submit the following:

- One or more files for the typechecker (source code); this should include all files for the scanner, the parser, and the typechecker. Make any necessary modifications to your code from Project 1. However, *do not use someone else's code for scanning, parsing, or printing* – use your own Project 1 code, modified as necessary.
- Makefile and Runfile, as in previous projects
- Text files ValidTests and InvalidTests
- Text file README.txt, containing
    - Your name on top
    - Additional details the grader may need to build and run your project
    - Design information: describe briefly the overall design. Point to parts of your code that implement the type checking. This text can be short, e.g., 1-2 paragraphs.
    - If you borrowed ideas or anything else from anywhere, describe briefly.

Login to *stdlinux*, in the directory that contains your files. Then use the following command:

submit c6341aa lab4 README.txt Makefile Runfile ValidTests InvalidTests file1 file2 …

Make sure that you submit only the files described above: do not submit files x.o, x.class, x.doc, etc.

If the time stamp on your electronic submission is **12:00 am on the next day or later**, you will receive 10% reduction per day, for up to three days. If your submission is later than 3 days after the deadline, it will **not** be accepted and you will receive zero points for this project. If you resubmit your project, this will override any previous submissions and only the latest submission will be considered – **resubmit at your own risk**. If the grader has problems with compiling or executing your program, she/he will e-mail you; you must respond within 48 hours to resolve the problem. Please check often your email accounts after submitting the project (for about a week or so) in case the grader needs to get in touch with you.

Grading

The grader will build you project using 'make' and will run it as shown in Runfile, using an extensive set of test cases. The grader will then read file README.txt and will check your code, and then will assign a grade. The grade will be primarily based on correctness on valid inputs and handling of invalid inputs.

Academic Integrity

The project you submit must be entirely your own work. Minor consultations with others are OK, but they should be at a very high level, without any specific details. The work on the project should be entirely your own: all design, programming, testing, and debugging should be done only by you, independently and from scratch. Sharing your code or documentation with others is not acceptable. Submissions that show excessive similarities (for code or for documentation) will be taken as evidence of cheating and dealt with accordingly; this includes any similarities with projects submitted in previous instances of this course.

Academic misconduct is an extremely serious offense with **severe** consequences. Details on academic integrity are available from the Committee on Academic Misconduct (*http://oaa.osu.edu/coamresources.html*). I strongly recommend that you check this URL. If you have any questions about university policies or what constitutes academic misconduct in this course, please contact me immediately.