Homework 6: Lectures 11 & 12

CS 440: Programming Languages and Translators, Spring 2020

Due Fri Feb 28, 11:59 pm

What to submit

Submit the Haskell program in a `*.hs` file. You can put the answers to the written problems in a separate file or as comments in the `*.hs` file (your choice). Please name the files something like `Doe_John_440_hw6.hs` or `pdf`, and if you have multiple files, put them into a folder named something like `Doe_John_440_hw6`, zip the folder, and submit the zip file.

Remember the new requirements: If you work alone, please say so in your submission. If you work in a group but aren't the person submitting the solution, then create a short file with the names of everyone in your group (including yourself), and submit that to Blackboard (in the homework 6 folder). These new requirements will make it easier for us to detect if someone forgot to put names down on the submission or didn't do the homework.

Problems [50 pts]

A. Lecture 11: Top-Down Parsing [10 points]

1. [5 points] Write out a leftmost derivation of `acdbecdebec` for the grammar below (*S* is the start symbol)

   $S \rightarrow \texttt{a}\ S\ \texttt{b}\ S\ \texttt{c} \mid \texttt{c}\ \texttt{d}\ S\ \texttt{d}\ S \mid \texttt{b}\ S\ \texttt{c} \mid \texttt{e}$

2. [5 points] The following grammar isn't LL(1). Modify the grammar to get one that generates the same language but is LL(1). $S \rightarrow \texttt{b}\ C \mid \texttt{b}\ D \quad C \rightarrow \texttt{c} \mid \texttt{c}\ C \quad D \rightarrow D\ \texttt{d} \mid \texttt{d}$

B. Lectures 11, 12: Recursive Descent Parsing

Programming Assignment (Lectures 11, 12) [40 pts]

- In previous homework, you've worked on a parser for expressions and on grammar rules for function applications. For this assignment, you're given a file `Skeleton/HW_06_Fcall.hs` and your job is to complete the program by replacing the stub sections with code so that it parses function calls and negative expressions. (There's also a stub for one of the tail nonterminals.)
- The parse tree `data` structure has been augmented to include `Call String [Ptree]` and `Negative Ptree`, Your code should use the grammar rules that appear in the program's comments.
- There's one syntax change from previous work: function calls must include at least one argument, so `f(x)` and `f(x,x)` are legal (for example), but `f()` should be rejected.
- The code that's included is written using the `bind` and `fails` routines from Lecture 12. For fullest credit, your code should use them also.

Grading Guide

- You'll need to complete the following routines:
  - [5 pts] Factor Tail.
  - [10 pts] `parse_id_or_call`
  - [10 pts] `parse_arguments`
  - [5 pts] `parse_arglist`
  - [5 pts] `parse_negative`
  - [5 pts] Complete all of the routines above, using only `bind` and `fails` to check the results of parses. (So no `case` expressions or `if` tests to check for `Nothing` vs `Just` *something* or for `Empty` vs nonempty parse trees.)
- Your submitted `*.hs` file should include the skeleton code plus all your new code. For testing, we should be able to start up ghci and `:load` your `*.hs` file into it and run the parser.
- There is also a file with some tests in it attached to this handout. It's reasonably thorough but that still doesn't guarantee total correctness of your program. It's called `HW_06_Fcall_Tests.hs` and should be loadable.

Hint: To parse an identifier or function call, you need to check for an identifier first before you can check for the parenthesized arguments. If the argument parse fails, you have a plain identifier; if it succeeds, you have a function call.

