Project 3 - Parser I
CS 4481

In this project you will implement the first part of a bottom-up parser, namely, item closure. We will be using the scanner implemented in the Antlr library.

1. Load the Parser project into your IDE (Netbeans/Eclipse). If you are using Netbeans, you will create a new project and then import all of the files from the p3 directory (drag and drop works fine).
2. Install Antlr
   (a) Download the Antlr jar file from antlr.org. Note: you may want to put this in your project directory (e.g., in a "lib" directory).
   (b) Add the Antlr jar to the project classpath:
      i. For Netbeans:
         A. Right-click on Parser project and choose "Properties"
         B. Choose "Libraries" on the left
         C. Click "Add JAR/Folder"
         D. Select *antlr-4.5.1-complete.jar* from wherever it is on your hard drive. The Quick Start tab on www.antlr.org should give you an idea of where it is.
      ii. For Eclipse:
         A. Right-click on Parser project and choose "Properties"
         B. Choose "Java Build Path" on the left menu and then choose the "Libraries" tab
         C. Click "Add External JARs"
         D. Select *antlr-4.5.1-complete.jar* from wherever it is on your hard drive.
3. There are a number of different classes and namespaces in the Parser project. You will work with code only in the parser namespace and you will modify only *Parser.java*.
4. To test your changes, you will uncomment tests in *main.java*. Do NOT make modifications to any other files. The only file you will submit for this project is *Parser.java*. See TODO comments in *main.java* and *Parser.java*.
5. Your parser will be tested using three grammars: *data/Simple.cfg*, *data/Paren.cfg* and *data/Expr.cfg*. See also their respective *data/*.token* files.
   Note on Paren.cfg: one production rule differs from what is in the book. See the following:
   - <u>*Paren.cfg*</u>: *pair* $\rightarrow$ *OPAREN list CPAREN*
   - <u>*The book*</u>: *pair* $\rightarrow$ *OPAREN pair CPAREN*

   The book simplifies the production rule so the closure sets aren't too big since it is demonstrating computing the tables by hand. It might be useful for you to change the production rule to match what is in the book *temporarily* for testing to make sure the results from your closure and other functions match what the book has. Just be sure to change it back or you won't be able to parse things like (()()).

Scoring

You will be graded on how many tests your code passes. The input files when grading will be similar to the test files you are using.