Homework 01 (Due: Friday, September 30, 2016, 11 : 59 : 00PM Central Time)

CSCE 322

1 Instructions

In this assignment, you will be required to scan, parse, and check the semantics of a file that encodes the state of a variation of Greater Than Sudoku. The definition of a properly formatted input file is given in Section 1.1.

You will be submitting one `.java` file and two `.g4` (ANTLR) files via web hand-in.

1.1 File Specification

- The file contains thre (3) labeled sections: **Spaces** , **Values** , and **Game** . Each section is enclosed by start and end tags (`%section%` and `$section`, respectively). The value of the section is set by the `=>` assignment operator.
- **Spaces** and **Values** are caret-separated (`^`) lists of spaces and symbols that appear between `%spaces%` and `$spaces` (and `%values%` and `$values`) tokens. Valid **Spaces** and **Values** are numerical symbols.
- **Game** contains three (3) two-dimensional arrays of space-separated entries that uses numeric symbols (and the `-`) to encode the state of the game. Rows will be ended with a `/n` and the **Game** will be begun with a `%game%` and ended with a `$game`. The first two-dimensonal array contains the actual values placed in the 16 spaces in the game. The second and third contain an encoding of the < and > signs that appear between each of the neighboring spaces in the game. The second encodes vertical >s (−1) and <s (1) (decreasing vs increasing), while the third encodes horizontal >s (−1) and <s (1).

An example of a properly formatted file is shown in Figure 1.

```
%section%   %title%<Values>   => %values%       4 ^   3 ^   4 ^   2 ^   3 ^   4 ^   3 ^   2 ^   2 ^   1 ^   2 ^   3 ^   2 $values
 $section
%section% %title%<Spaces>     =>    %spaces%       1 ^  15 ^   5 ^   7 ^   2 ^  12 ^   8 ^  14 ^   3 ^  16 ^  11 ^   4 ^  13 $spaces
 $section
%section%     %title%<Game>    =>    %game%
   -     -     4     -/n
   -     1     3     -/n
   -     -     -     -/n
   -     -     -     -
 $game
 %game%
   1    -1    -1     1/n
  -1     1    -1     1/n
  -1    -1     1     1
 $game
 %game%
  -1     1    -1/n
  -1     1    -1/n
   1    -1     1/n
   1    -1     1
 $game
 $section
```

Figure 1: A properly formatted Greater Than Sudoku (gts) encoding

The assignment is made up of two parts: scanning the text of the input file and parsing the information contained in the input file.

1.2 Scanning

Construct a combined grammar in a `.g4` file that ANTLR can use to scan a supplied Greater Than Sudoku encoding. The logic in this file should be robust enough to identify tokens in the encoding and accurately process any correctly formatted encoding. The rules in your `.g4` file will be augmented with actions that display information about the input file. An example of that output is specified in Section 2.

The purpose of the scanner is to extract tokens from the input and pass those along to the parser. For the Greater Than Sudoku encoding, the types of tokens that you will need to consider are given in Table 1.

Type	Form
Section Beginning	`%section%`
Section Ending	`$section`
Section Title	`%title%<Spaces>`, `%title%<Values>` and `%title%<Game>`
Assign Value	`=>`
Space Symbol	One or more Numerical Symbols
Value Symbol	One or more Numerical Symbols
Game Symbol	`-` or One or more Numerical symbols
Numerical Symbol	`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, or `-`
Row Ending	`/n`
Game Beginning	`%game%`
Game Ending	`$game`
List Beginning	`%spaces%` or `$spaces`
List Ending	`%values%` or `$values`
White Space (to be ignored)	spaces, tabs, newlines

Table 1: Tokens to Consider

1.2.1 Invalid Encodings

For invalid Greater Than Sudoku encodings, the output `ERR: T in line L.` should display. `T` would be the symbol read and `L` would be the line of input where the symbol was read. Your scanner should stop scanning the file after an unrecognized token is found.

1.3 Parsing

Construct a combined grammar in a `.g4` file that ANTLR can use to parse a supplied Greater Than Sudoku encoding. In addition to the rules for scanning, there are several parsing rules:

- Each section appears once and only once. The sections may appear in either **Spaces** /**Values** /**Game** or **Values** /**Spaces** /**Game** order.
- There must be more than two (2) rows in a valid **Game** .
- There must be more than two (2) columns in a valid **Game** .

  You may assume that each row has the same number of columns, and each column has the same number of rows.

  You may assume that the visible game always comes before the grid of row inequalities and the grid of column inequalities.
- There must be more than three (3) spaces in the **Spaces** section and more than three values in the **Values** section.

The semantics of a properly formatted Greater Than Sudoku encoding are:

1. The number of rows/columns must be equal and a perfect square
2. The number of numerical symbols already placed in the **Game** must not exceed 25% of all spaces in the **Game** .
3. **Extra Credit (10 points or Honors contract)**: The **Game** must still be satisfy all the contraints of Greater than Sudoku (each of the numbers 1 through n (number of rows/columns) appears at most once in each row/column and each $\sqrt{n} \times \sqrt{n}$ cluster of spaces, and all $<$ and $>$ signs are obeyed.

2 Output

2.1 Scanner

Your `.g4` file should produce output for both correctly formatted files and incorrectly formatted files. For the correctly formatted file in Figure 1, the output would have the form of the output presented in Figure 2

```
Start Section: %section%
Label: %title%<Values>
Assign: =>
Start Values: %values%
Numerical Symbol: 4
...
Numerical Symbol: 2
Cease Values: $values
Cease Section: $section
Start Section: %section%
Label: %title%<Spaces>
Assign: =>
Start Spaces: %spaces%
Numerical Symbol: 1
...
Numerical Symbol: 13
Cease Spaces: $spaces
Cease Section: $section
Start Section: %section%
Label: %title%<Game>
Assign: =>
Start Values: %game%
Open Space: -
Open Space: -
Numerical Symbol: 4
Open Space: -
Cease Row: /n
Open Space: -
...
Numerical Symbol: 1
Cease Values: $game
Cease Section: $section
Cease File
```

Figure 2: Truncated Output of Scanner for File in Figure 1

For a correctly formatted file in Part 2, the output would be: `You need to fill s spaces.` where `s` is the number of unfilled spaces in the **Game** . For the file in Figure 1, the output would be

`You need to fill 13 spaces..`

2.1.1 Invalid Syntax & Semantics in Parsing

For invalid encodings in Part 2, a message describing the error should be displayed. For a syntax error (violation of the syntax rules), the output
`ERR: T in line L.` should be displayed, where `L` is the line number where the unrecognized token `T` was found. For that error, the parser should stop processing the file. For a semantic rule violation, the output `SEMANTIC ERROR: Rule R.` should be displayed, where `R` is the number of the rule (from List 1.3) that was violated, but parsing should continue.

Syntax errors in Part 2 should be reported in the `syntaxError` method of `csce322hw001pt02error.java`.

3 Naming Conventions

The ANTLR file for the first part of the assignment should be named `csce322hw001pt01.g4`. The ANTLR file for the second part of the assignment should be named `csce322hw001pt02.g4`. Both grammars should contain a start rule named `greaterThanSudoku`. The Java file for the second part of the assignment should be named `csce322hw001pt02error.java`.

4 webgrader

The webgrader is available for this assignment. You can test your submitted files before the deadline by submitting them on webhandin and going to http://cse.unl.edu/~cse322/grade, choosing the correct assignment and entering your `cse.unl.edu` credentials

The script should take approximately 2 minutes to run and produce a PDF.

4.1 The Use of `diff`

Because Part 1 of this assignment only depends on the symbols in the file, the order in which they are displayed should not be submission dependent. Therefore, `diff` will be used to compare the output of a particular submission against the output of the solution implementation.

5 Point Allocation

Component	Points
Part 1	35
Part 2	65
Total	100

6 External Resources

ANTLR
Getting Started with ANTLR v4
ANTLR 4 Documentation
Overview (ANTLR 4 Runtime 4.5 API)

7 Commands of Interest

```
alias antlr4='java -jar /path/to/antlr-4.5.3-complete.jar'
alias grun='java org.antlr.v4.gui.TestRig'
export CLASSPATH="/path/to/antlr-4.5.3-complete.jar:$CLASSPATH"
antlr4 /path/to/csce322hw01part0#.g4
javac -d /path/for/.classfiles /path/to/csce322hw01part0#*.java
java /path/of/.classfiles csce322hw01part02driver /path/to/inputfile
grun csce322hw01pt0# greaterThanSudoku -gui
grun csce322hw01pt0# greaterThanSudoku -gui /path/to/inputfile
grun csce322hw01pt0# greaterThanSudoku
grun csce322hw01pt0# greaterThanSudoku /path/to/inputfile
```