Homework 01 (Due: Friday, February 21, 2020, 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 Peg Solitaire. 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 two (2) labeled sections: **Moves** and **Game** . Each section is enclosed by start and end tags (`>>` and `<<`, respectively).
- **Moves** is an comma-separated (`,`) list of directions that appear between `^` and `$` tokens. Valid **Moves** are `u`, `d`, `l`, and `r`.
- **Game** contains a two-dimensional array of space-separated entries that uses numeric symbols (and `-` and `x`) to encode the state of the game. Rows will be ended with a `*` and the **Game** will be begun with a `{` and ended with a `}`. The two-dimensonal array contains numbers where that player (for example, `1` for Player 1) has to move a piece into on their next move, `x` for a piece left on the board, and `-` for an empty space.

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

```
  !game    >>
 {
x x x x x x x x x x x 3 - x - x x x x *
x x x x x x x 2 x x x x x x x - 1 x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x 4 x x x *
x x x x x x x x x x x x x x x - x x x *
x x x x x x x x x x x x x x x x - x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x *
x x x x x x x x x x x x x x x x x x x
  }
   <<
    !moves    >>
 ^   r , d , r , r , l , d , l , r , d , d , r , u , d , r , l , u $
     <<
```

Figure 1: A properly formatted Extreme Peg Solitaire (eps) 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 Extreme Peg Solitaire 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 Extreme Peg Solitaire encoding, the types of tokens that you will need to consider are given in Table 1.

Type	Form
Section Beginning	`>>`
Section Ending	`<<`
Section Title	`!game` and `!moves`
Move Symbol	`u`, `d`, `l`, or `r`
Game Symbol	`-`, `x`, or One or more Numerical Symbols
Numerical Symbol	`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, or `9`
Row Ending	`*`
Game Beginning	`{`
Game Ending	`}`
Moves Beginning	`^`
Moves Ending	`$`
White Space (to be ignored)	spaces, tabs, newlines

Table 1: Tokens to Consider

1.2.1 Invalid Encodings

For invalid Extreme Peg Solitaire encodings, the output `SYNTAX ERROR IN LINE L` should display. `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 Extreme Peg Solitaire 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 **Moves** /**Game** or **Game** /**Moves** order.
- There must be at least ten (10) rows in a valid **Game** .
- There must be at least ten (10) 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.
- There must be at least five (5) moves in the **Moves** section.

The semantics of a properly formatted Connect Four encoding are:

1. The **Game** must have 2 to 4 players
2. The **Game** must contain no more than 40% empty spaces
3. The **Moves** section must have at least one of each move (`u`, `d`, `l`, and `r`)
4. **Extra Credit (**10 **points or Honors contract)**: No player may be completely surrounded (up, down, left, and right) by empty spaces

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

```
game Section
Begin the Section
Start the Game
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: 3
Space: Empty
Space: x
Space: Empty
Space: x
Space: x
Space: x
Space: x
End the Row
Space: x
...
Space: x
End the Row
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
Space: x
End the Game
End the Section
moves Section
Begin the Section
Begin the List
Move: r
...
Move: u
End the List
End the Section
End the File
```

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

For a correctly formatted file in Part 2, the output would be: `There are s empty spaces on the board` where `s` is the number of empty spaces in the **Game** . For the file in Figure 1, the output would be

```
There are 5 empty spaces on the board
```

2.1.1 Invalid Syntax & Semantics in Parsing

For invalid encodings in Part 2, the message `SYNTAX ERROR IN LINE L` should be displayed where `L` is the line number where the error occurred. For a semantic rule violation, the output `SEMANTIC ERROR P` should be displayed, where `P` 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 `csce322a01partt02error.java`.

3 Naming Conventions

The ANTLR file for the first part of the assignment should be named `csce322a01part01.g4`. The ANTLR file for the second part of the assignment should be named `csce322a01part02.g4`. Both grammars should contain a start rule named `extremePegSolitaire`. The Java file for the second part of the assignment should be named `csce322a01part02error.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. In Part 2, the output is sorted and the unique lines extracted, so the order and number of times a semantic error is reported will not affect the diff.

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 4.8 API)

7 Commands of Interest

```
alias antlr4='java -jar /path/to/antlr-4.8-complete.jar'
alias grun='java org.antlr.v4.gui.TestRig'
export CLASSPATH="/path/to/antlr-4.8-complete.jar:$CLASSPATH"
antlr4 /path/to/csce322a01part0#.g4
javac -d /path/for/.classfiles /path/to/csce322a01part0#*.java
java /path/of/.classfiles csce322a01part02driver /path/to/inputfile
grun csce322a01part0# extremePegSolitaire -gui
grun csce322a01part0# extremePegSolitaire -gui /path/to/inputfile
grun csce322a01part0# extremePegSolitaire
grun csce322a01part0# extremePegSolitaire /path/to/inputfile
```