Assignment 01 (Due: Wednesday, October 1, 2014, 11 : 59 : 00PM)

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 the game Chutes and Ladders. 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, along with a `README.txt` file via web hand-in.

1.1 File Specification

- The file contains four (4) labeled sections: **Positions** , **Chutes** , **Ladders** , and **GameBoard** . Each section appears once (and only once) in each file. Each section is enclosed by start and end tags (`<=` and `=>`, respectively). The value of the section is set by the `<-` assignment operator.
- **Positions** , **Chutes** , and **Ladders** are lists of values that appear as comma-separated lists in curly braces (`{` and `}`).
- **GameBoard** is a two-dimensional array of space-separated entries that uses the same alphabet as **Positions** , **Chutes** , and **Ladders** to encode the next board location that a piece on that space would go to. Rows will be ended with a `^` and the **GameBoard** will be ended with a `]` (and is begun with a `[`).

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

```
<= Positions <- { 1 , 1 , 1 } =>
<= Chutes <- { 4 , 24 , 27 } =>
<= Ladders <- { 9 , 16 , 28 } =>
<= GameBoard <- [
   0, 30, 30, 10, 27^
  22, 23, 24,  6, 26^
  21, 20, 19, 18, 27^
  12, 13, 14, 15, 16^
  11, 13,  9,  8,  7^
   2,  3,  4,  3,  6
] =>
```

Figure 1: A properly formatted Chutes and Ladders 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 Chutes and Ladders 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 Chutes and Ladders encoding, the types of tokens that you will need to consider are given in Table 1.

Type	Form
Beginning of Section	`<=`
End of Section	`=>`
Type of Section	`Board`, `Ladders`, `Chutes`, and `Positions`
Value Assignment	`<-`
Numerical Symbol	`0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`
Positions /**Chutes** /**Ladders** Symbol	1 or more Numerical Symbols
GameBoard Symbol	**Chutes** /**Ladders** Symbol
Row Separator	`^`
Beginning of Board	`[`
End of Board	`]`
Beginning of List	`{`
End of List	`}`
White Space (to be ignored)	spaces, tabs, newlines

Table 1: Tokens to Consider

1.2.1 Invalid Encodings

For invalid Chutes and Ladders encodings, the output `T in Line L is Not a Recognized Token` 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 Chutes and Ladders encoding. In addition to the rules for scanning, there are several parsing rules:

- Each section appears once and only once. The **Positions** section comes first. **Chutes** and **Ladders** sections must be the second and third sections in a properly formatted file, however, they may appear in either **Chutes** /**Ladders** or **Ladders** /**Chutes** order. The fourth section of a properly formatted file must be the **GameBoard** .
- There must be more than one player (whose locations on the board is encoded in **Positions**)
- There must be at least one chute and one ladder
- The board must contain more than one row and more than one column

The semantics of a properly formatted Chutes and Ladders encoding are:

1. Each **Position** (and each next location on the board) must be greater than or equal to zero (0) and less than or equal to the number of spaces on the **GameBoard** .
2. The number of players must be greater than or equal to two (2) and less than or equal to four (4).
3. The number of chutes on the **GameBoard** must be greater than two (2) and less than eleven (11).
4. The number of ladders on the **GameBoard** must be greater than two (2) and less than eleven (11).
5. The number of rows on the **GameBoard** must be greater than four (4) and less than twenty (20).
6. The number of columns on the **GameBoard** must be greater than four (4) and less than twenty (20).
7. **Extra Credit (5 points)**: Chutes cannot make up more than 30% of the spaces on the board and ladders cannot make up more than 40% of the locations on the board.

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 (truncated) output presented in Figure 2

```
Start of Section: <=
Name of Section: Positions
Assignment of Value: <-
Start of List: {
Chute/Ladder/Space: 1
Chute/Ladder/Space: 1
Chute/Ladder/Space: 1
End of List: }
End of Section: =>
Start of Section: <=
Name of Section: Chutes
Assignment of Value: <-
Start of List: {
Chute/Ladder/Space: 4
Chute/Ladder/Space: 24
Chute/Ladder/Space: 27
End of List: }
End of Section: =>
Start of Section: <=
Name of Section: Ladders
Assignment of Value: <-
Start of List: {
Chute/Ladder/Space: 9
Chute/Ladder/Space: 16
Chute/Ladder/Space: 28
End of List: }
End of Section: =>
Start of Section: <=
Name of Section: GameBoard
Assignment of Value: <-
Start of Board: [
Chute/Ladder/Space: 0
Chute/Ladder/Space: 30
Chute/Ladder/Space: 30
Chute/Ladder/Space: 10
Chute/Ladder/Space: 27
End of Row: ^
Chute/Ladder/Space: 22
...
Chute/Ladder/Space: 6
End of Board: ]
End of Section: =>
End of File
```

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

For a correctly formatted file in Part 2, the output would be: `The board has c chutes and l ladders` where `c` is the total number of chutes on the **GameBoard** and `l` is the number of ladders. For the file in Figure 1, the output would be `The board has 3 chutes and 3 ladders`.

2.1.1 Invalid Syntax & Semantics in Parsing

For invalid Chutes and Ladders encodings in Part 2, a message describing the parsing error should be displayed. For an unrecognized token (not in the alphabet of tokens), the output `T in Line L is an Unrecognized Token` should be displayed, where `L` is the line number where the unrecognized token `T` was found. For a syntax error (valid token, out of place), the output

`T in Line L is an Invalid Token` should be displayed, where `L` is the line number where the invalid token `T` was found. On both of those errors, the parser should stop processing the file. For a semantic violation, the output `Semantic Error: Rule R Violated` should be displayed, where `R` is the number of the rule (from List 1.3) that was violated, but parsing should continue.

3 Naming Conventions

The ANTLR file for the first part of the assignment should be named `csce322as01pt01.g4`. The ANTLR file for the second part of the assignment should be named `csce322as01pt02.g4`. Both grammars should contain a start rule named `chutesNladders`. The Java file for the second part of the assignment should be named `csce322as01pt02dr.java`.

The `.java` file should accept a command-line argument (filename). Include a `README.txt` file with your submission. Your `README.txt` file should include assumptions that you made about the problem and sources you consulted during the completion of this assignment.

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.

Part 2 is more submission-dependent. When semantic errors are reported and what order they are reported in will affect a submission's output. This is taken into account (and is an assumption that should be reported in `README.txt`).

5 Point Allocation

Component	Points
Part 1	30
Part 2	60
Documentation (`README.txt` with assumptions that you made to handle cases that did not arise in the specifications and any sources that you consulted while completing the assignment)	10
Total	100

6 External Resources

ANTLR
Getting Started with ANTLR v4

ANTLR 4 Documentation
Overview (ANTLR 4 Runtime 4.4 API)

7 Commands of Interest

```
alias antlr4='java -jar /path/to/antlr-4.4-complete.jar'
alias grun='java org.antlr.v4.runtime.misc.TestRig'
export CLASSPATH="/path/to/antlr-4.4-complete.jar:$CLASSPATH"
antlr4 /path/to/csce322as01pt0#.g4
javac -d /path/for/.classfiles /path/to/csce322as01pt0#*.java
java /path/of/.classfiles
grun csce322as01p0# chutesNladders -gui
grun csce322as01p0# chutesNladders -gui /path/to/inputfile
grun csce322as01p0# chutesNladders
grun csce322as01p0# chutesNladders /path/to/inputfile
```

8 `webgrader` Script

```
#!/bin/bash

# preformatted HTML text
echo "<p>"

# some basic information
USER=$1
ASSIGNMENT=$2
BASE=csce322a${ASSIGNMENT}
DIR=webhandin/$ASSIGNMENT/CSCE322/$USER
LOCALDIR=${BASE}/$USER
SOL=${BASE}/Solution
WEB=public_html/reports/${ASSIGNMENT}/${USER}

# the tools we'll use
diff='/usr/bin/diff'
grep='/usr/bin/grep'
matlab='/usr/local/bin/matlab'
pdflatex='/usr/bin/pdflatex'
javac='/usr/bin/javac -J-Xmx2048m'
java='/usr/bin/java -Xmx2048m'
antlr4="$java -jar $LOCALDIR/antlr-4.4-complete.jar"
grun="$java org.antlr.v4.runtime.misc.TestRig"
ps2pdf='/usr/bin/ps2pdf'

# the name of the produced PDF
pdfname="${USER}_${ASSIGNMENT}"

# don't modify handin files
rm -rf $LOCALDIR &> output.garbage
cp -r $DIR $LOCALDIR
chmod 770 $LOCALDIR/*
chmod 770 $LOCALDIR

# the beginnings of the LaTeX document
cat preamble.tex > $LOCALDIR/output.tex
echo "\input{csce322a${ASSIGNMENT}/rubric}" >> $LOCALDIR/output.tex
echo '\setcounter{tocdepth}{4}' >> $LOCALDIR/output.tex
echo '\tableofcontents' >> $LOCALDIR/output.tex

# removing auto-generated files
rm $LOCALIDR/*.output output.* $LOCALDIR/*.err $LOCALDIR/*.time &> output.garbage

# set limits
#ulimit -t 60 # time

echo '\chapter{Metadata}' >> $LOCALDIR/output.tex
echo '\section{Submitted Files}' >> $LOCALDIR/output.tex
$grep -e $USER webhandin/$ASSIGNMENT/record > $LOCALDIR/handin.time
echo "\lstinputlisting[title=handin.time,numbers=left]{$LOCALDIR/handin.time}" >>
    $LOCALDIR/output.tex
echo '\section{webgrader Runs}' >> $LOCALDIR/output.tex
$grep -e 2014[^A-Z]*${USER} public_html/grade/grade.log > $LOCALDIR/webgrader.time
echo "\lstinputlisting[title=webgrader.time,numbers=left]{$LOCALDIR/webgrader.time
    }" >> $LOCALDIR/output.tex

echo '\chapter{\texttt{README.txt}}' >> $LOCALDIR/output.tex
if [ -e $LOCALDIR/README.txt ]
then
    echo "\lstinputlisting[title=README.txt]{$LOCALDIR/README.txt}" >> $LOCALDIR/
    output.tex
else
    echo '\texttt{README.txt} Not Found' >> $LOCALDIR/output.tex
fi

cp $BASE/antlr-4.4-complete.jar $LOCALDIR/

echo '\chapter{\texttt{csce322as01pt01.g4}}' >> $LOCALDIR/output.tex
if [ -e $LOCALDIR/csce322as01pt01.g4 ]
then
    # add the ANTLR jar and the user's folder to the Java classpath
    export CLASSPATH="$BASE/antlr-4.4-complete.jar:$LOCALDIR:$LOCALDIR/01"

    # make a directory for Part 01
    mkdir $LOCALDIR/01 > output.garbage

    # compile it and redirect errors
    $antlr4 -o $LOCALDIR/01 $LOCALDIR/csce322as01pt01.g4 &> $LOCALDIR/01/ANTLR.err

    # if it compiled successfully
    if [ -e $LOCALDIR/01/csce322as01pt01Parser.java ]
    then
        # compile the .java files, place them in LOCALDIR, and redirect errors
        $javac -d $LOCALDIR/01 $LOCALDIR/01/*.java &> $LOCALDIR/01/javac.err

        # if the Java files compiled successfully
        if [ -e $LOCALDIR/01/csce322as01pt01Parser.class ]
        then
            cp $BASE/*.m $LOCALDIR/01/
            sed "s@DIRECTORY@$LOCALDIR/01@g" $BASE/makeTests.m > $LOCALDIR/01/
    makeTests.m
            sed "s@TEST@10@g;s@DIRECTORY@$LOCALDIR/01@g;s@PART@01@g" $BASE/
    printANTLR.m > $LOCALDIR/01/printANTLR.m
            $matlab < $LOCALDIR/01/makeTests.m > output.garbage

            # run the tests for this part
            for file in `seq 1 10`
            do
                filename=`printf "part01test%02d.cal" $file`
                $grun csce322as01pt01 chutesNladders -ps $LOCALDIR/01/`basename
    $filename .cal`.ps $LOCALDIR/01/$filename &> $LOCALDIR/01/`basename $filename .
    cal`.output
                $ps2pdf -dEPSCrop -dNOSAFER $LOCALDIR/01/`basename $filename .cal
    `.ps $LOCALDIR/01/`basename $filename .cal`.pdf &> output.garbage
            done

            # get the solutions for this part
            # add the ANTLR jar and the user's folder to the Java classpath
            export CLASSPATH="$BASE/antlr-4.4-complete.jar:$SOL"

            for file in `seq 1 10`
            do
                filename=`printf "part01test%02d.cal" $file`
                $grun csce322as01pt01 chutesNladders $LOCALDIR/01/$filename 1>
    $LOCALDIR/01/`basename $filename .cal`.solution 2> $LOCALDIR/01/`basename
    $filename .cal`.err
                $diff $LOCALDIR/01/`basename $filename .cal`.output $LOCALDIR/01/`
    basename $filename .cal`.solution > $LOCALDIR/01/`basename $filename .cal`.diff
     # output should be identical
                echo "\section{$filename}" >> $LOCALDIR/output.tex
                echo '\subsection{Diff}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.diff]{
    $LOCALDIR/01/`basename $filename .cal`.diff}" >> $LOCALDIR/output.tex # show
    the diff result
                echo '\subsection{Input File}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=$filename]{$LOCALDIR/01/$filename}"
    >> $LOCALDIR/output.tex # show the file
                echo '\subsection{Submission Output}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.output]{
    $LOCALDIR/01/`basename $filename .cal`.output}" >> $LOCALDIR/output.tex # show
    the submission's output
                echo '\subsection{Sample Output}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.output]{
    $LOCALDIR/01/`basename $filename .cal`.solution}" >> $LOCALDIR/output.tex #
    show the submission's output
                echo '\subsection{Parse Tree}' >> $LOCALDIR/output.tex
                sed "s@PATH@$LOCALDIR/`basename $filename .cal`@g" $BASE/tree.tex
    >> $LOCALDIR/output.tex
                echo '\subsection{\texttt{stderr}}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.err]{
    $LOCALDIR/01/`basename $filename .cal`.err}" >> $LOCALDIR/output.tex # show the
     submission's output
            done
        else
            echo '\section{\texttt{javac} Errors}' >> $LOCALDIR/output.tex
            echo "\lstinputlisting[title=javac.err]{$LOCALDIR/01/javac.err}" >>
    $LOCALDIR/output.tex
        fi
    else
        echo "\section{ANTLR Errors}" >> $LOCALDIR/output.tex
        echo "\lstinputlisting[title=ANTLR.err]{$LOCALDIR/01/ANTLR.err}" >>
    $LOCALDIR/output.tex
    fi
    # put the source code in the report
    echo '\section{Source Code}' >> $LOCALDIR/output.tex
    echo "\lstinputlisting[numbers=left,title=csce322as01pt01.g4]{$LOCALDIR/
```

```
    csce322as01pt01.g4}" >> $LOCALDIR/output.tex
else
    echo '\texttt{csce322as01pt01.g4} Not Found' >> $LOCALDIR/output.tex
fi

echo '\chapter{\texttt{csce322as01pt02.g4}}' >> $LOCALDIR/output.tex
if [ -e $LOCALDIR/csce322as01pt02.g4 ]
then
    # add the ANTLR jar and the user's folder to the Java classpath
    export CLASSPATH="$BASE/antlr-4.4-complete.jar:$LOCALDIR:$LOCALDIR/02"

    # make a directory for Part 01
    mkdir $LOCALDIR/02 > output.garbage

    # compile it and redirect errors
    $antlr4 -o $LOCALDIR/02 $LOCALDIR/csce322as01pt02.g4 &> $LOCALDIR/02/ANTLR.err

    # if it compiled successfully
    if [ -e $LOCALDIR/02/csce322as01pt02Parser.java ]
    then
        cp $LOCALDIR/csce322as01pt02dr.java $LOCALDIR/02/ &> output.garbage

        # compile the .java files, place them in LOCALDIR, and redirect errors
        $javac -d $LOCALDIR/02 $LOCALDIR/02/*.java &> $LOCALDIR/02/javac.err

        # if the Java files compiled successfully
        if [ -e $LOCALDIR/02/csce322as01pt02Parser.class ]
        then
            cp $BASE/*.m $LOCALDIR/02/
            sed "s@DIRECTORY@$LOCALDIR/02@g" $BASE/makeTests.m > $LOCALDIR/02/
    makeTests.m
            sed "s@TEST@15@g;s@DIRECTORY@$LOCALDIR/02@g;s@PART@02@g" $BASE/
    printANTLR.m > $LOCALDIR/02/printANTLR.m

            $matlab < $LOCALDIR/02/makeTests.m > output.garbage

            # add the ANTLR jar and the user's folder to the Java classpath
            export CLASSPATH="$BASE/antlr-4.4-complete.jar:$LOCALDIR/02"

            # run the tests for this part
            for file in `seq 1 15`
            do
                filename=`printf "part02test%02d.cal" $file`
                $java csce322as01pt02dr $LOCALDIR/02/$filename 1> $LOCALDIR/02/`
    basename $filename .cal`.output 2> $LOCALDIR/02/`basename $filename .cal`.err
                $grun csce322as01pt02 chutesNladders -ps $LOCALDIR/02/`basename
    $filename .cal`.ps $LOCALDIR/02/$filename &> output.garbage
                $ps2pdf -dEPSCrop -dNOSAFER $LOCALDIR/02/`basename $filename .cal
    `.ps $LOCALDIR/02/`basename $filename .cal`.pdf &> output.garbage
            done

            # get the solutions for this part
            # add the ANTLR jar and the user's folder to the Java classpath
            export CLASSPATH="$BASE/antlr-4.4-complete.jar:$SOL"

            for file in `seq 1 15`
            do
                filename=`printf "part02test%02d.cal" $file`
                $java csce322as01pt02dr $LOCALDIR/02/$filename > $LOCALDIR/02/`
    basename $filename .cal`.solution
                $diff $LOCALDIR/02/`basename $filename .cal`.output $LOCALDIR/02/`
    basename $filename .cal`.solution > $LOCALDIR/02/`basename $filename .cal`.diff
     # output should be identical
                echo "\section{$filename}" >> $LOCALDIR/output.tex
                echo '\subsection{Diff}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.diff]{
    $LOCALDIR/02/`basename $filename .cal`.diff}" >> $LOCALDIR/output.tex # show
    the diff result
                echo '\subsection{Input File}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=$filename]{$LOCALDIR/02/$filename}"
    >> $LOCALDIR/output.tex # show the file
                echo '\subsection{Submission Output}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.output]{
    $LOCALDIR/02/`basename $filename .cal`.output}" >> $LOCALDIR/output.tex # show
    the submission's output
                echo '\subsection{Sample Output}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.output]{
    $LOCALDIR/02/`basename $filename .cal`.solution}" >> $LOCALDIR/output.tex #
    show the submission's output
                echo '\subsection{Parse Tree}' >> $LOCALDIR/output.tex
                sed "s@PATH@$LOCALDIR/`basename $filename .cal`@g" $BASE/tree.tex
    >> $LOCALDIR/output.tex
                echo '\subsection{\texttt{stderr}}' >> $LOCALDIR/output.tex
                echo "\lstinputlisting[title=`basename $filename .cal`.err]{
    $LOCALDIR/02/`basename $filename .cal`.err}" >> $LOCALDIR/output.tex # show the
     submission's output
            done
        else
            echo '\section{\texttt{javac} Errors}' >> $LOCALDIR/output.tex
            echo "\lstinputlisting[title=javac.err]{$LOCALDIR/02/javac.err}" >>
    $LOCALDIR/output.tex
        fi
    else
        echo "\section{ANTLR Errors}" >> $LOCALDIR/output.tex
        echo "\lstinputlisting[title=ANTLR.err]{$LOCALDIR/02/ANTLR.err}" >>
    $LOCALDIR/output.tex
    fi
    # put the source code in the report
    echo '\section{Source Code}' >> $LOCALDIR/output.tex
    echo "\lstinputlisting[numbers=left,title=csce322as01pt02dr.java]{$LOCALDIR/
    csce322as01pt02dr.java}" >> $LOCALDIR/output.tex
    echo "\lstinputlisting[numbers=left,title=csce322as01pt02.g4]{$LOCALDIR/
    csce322as01pt02.g4}" >> $LOCALDIR/output.tex
else
    echo '\texttt{csce322as01pt02.g4} Not Found' >> $LOCALDIR/output.tex
fi

echo '\end{document}' >> $LOCALDIR/output.tex

# compile the PDF
$pdflatex -interaction batchmode $LOCALDIR/output.tex > output.garbage
$pdflatex -interaction batchmode $LOCALDIR/output.tex > output.garbage

# move the log file to the user's folder
cp output.log logs/$pdfname.log

# if the PDF was generated successfully
if [ -e output.pdf ]
```

```
then
    mkdir -p $WEB
    chmod a+x $WEB
    sed "s/USER/$USER/g" htaccess > $WEB/.htaccess
    chmod 644 $WEB/.htaccess
    mv output.pdf $WEB/$pdfname.pdf
    chmod 644 $WEB/$pdfname.pdf
    echo "PDF can be found at <a href=https://cse.unl.edu/~cse322/reports/
   $ASSIGNMENT/$USER/$pdfname.pdf>https://cse.unl.edu/~cse322/reports/$ASSIGNMENT/
   $USER/$pdfname.pdf</a>"
else
    echo "PDF was not generated successfully"
fi

# clean up
rm output.*
echo "</p>"
```