CSCE 322 - Fall 2010

Assignment 2: Lexing, Parsing, and checking some Semantics

Due Date: **Friday, September 24th before class starts**

Task

For this assignment you will use lex and yacc to scan and parse a file representing a collection of toy railroad pieces (imagine something like Thomas The Train tracks.) The file's specification is as follows:

- The file is made of tags and has a structure similar to that of an XML file. The tags form a hierarchy. The top of our hierarchy is the <track> tag, which is closed at the end of the file with a corresponding </track> tag. Tags that can have tags inside of them use the same opening and closing delimiters. In this project we have the following tags: <piece> and <station>.
- Some tags can have associated attributes. Each attribute is delimited by <- and -> and consists of the attribute name followed by an equal sign, a double quotation, the attribute value, and an ending double quotation mark. Attribute values can either be strings or integers. Strings will consist of alphanumeric characters. Integers will only have digits.
- A track tag may have a *name* attribute whose value is of string type.
- A <piece> tag must have a *shape, size, cost, and quantity* attributes. The attributes may appear in any order. *shape* must be one of the following strings: "straight", "curve", or "junction". *size*, *cost*, and *quantity* must all be assigned integer values.
- A <piece> *may* have **one** <station> tag inside of it. This new tag has an attribute *kind*, which must be one of the following strings: "police", "fire", or "depot", and an attribute *location* which indicates where it is located in the piece (ranging in value from 1 to piece's size).

Here is a sample file:

```
<track>
        <-name = "My own" ->
        <piece>
                <- shape ="curve" ->
                <- size = "2" ->
                <- quantity=  "4" ->
                <- cost  ="3" ->
        </piece>
        <piece>
                <- shape ="straight" ->
                <- size = "3" -> <- quantity=  "2"->     <- cost  ="2" ->
        </piece>
        <piece>
                <- shape ="straight" -> <- size = "2" -> <- quantity=  "2"->
                <- cost  ="1" ->
                <station>
                        <- kind = "police"-> <- location = "1" ->
                </ station>
        </piece>
        ...
</track>
```

This assignment has two parts. The first calls for creating a scanner that identifies the parts of the file and catches lexical errors. The second part will integrate the scanner with a parser to check for syntax errors and apply some semantic rules. Both programs should as argument the name of the track configuration file to process.

Part 1:

Develop a scanner for a configuration file that follows the previous specifications. Remember that the scanner's only responsibility is to split the input string into tokens.

Below is a list of categories that you will need to use in your scanner.

Category	Tokens in this category
Start Tag	<track>, <piece>, <station>
End Tag	</track>, </piece>, </station>
Start Attribute	<-
End Attribute	->
Attribute	name, shape, size, quantity, cost, kind, location
Assignment	=
Number	Any non-negative integer within double quotes
String	Any sequence of alphanumeric characters (that isn't a Number)

White spaces, tabs, and newlines should be ignored.

If an invalid token is read, the error message "Invalid token: T" should be displayed, where T is the first invalid character that was read. After an error occurs, the program should stop processing the file.

Your lex will need to output: matched category, colon, matched token, and a line return.

For example, for the previous example file the output would be:

```
Start Tag: <track>
Start Attribute: <-
Attribute: name
Assignment: =
String: "My own"
End Attribute: ->
Start Tag: <piece>
Start Attribute: <-
Attribute:shape
Assignment: =
String: "curve"
End Attribute: ->
...
```

Part 2:

Build a program using lex and yacc that checks for the syntax and semantics of the railroad configuration file. The syntax should be inferred from the specifications provided earlier in this document. The semantic rules follow.

Semantic rules for the XML configuration file:

1. A piece's quantity and size must be at least 1.
2. A junction must always have a size of 1.
3. A station may only be paired with a track whose shape is "straight".
4. The total number of curves must be at least 4 and a multiple of 2.
5. The total number of pieces in the collection must be less than 100.
6. Each kind of piece (where kind is defined by a piece's shape, size, cost, and the location and type of station if it has one) should appear only once per file (it is wrong for a piece with the same attributes except for the quantity to appear twice in a file)

As soon as an error is encountered, the program should print out the line number in which it occurred and the type of error:

- "Lexer: T" where T is the unrecognized token
- "Syntax: T" where T is the token that broke the grammar
- "Semantic: R", where R is the number corresponding to the rule that was broken as per the enumeration above.

If no errors are found, the program should display: "Success: N track pieces collected", where N is the total number of pieces of track parsed (sum of the "quantity" attribute of all pieces).

Note: to complete Part 2 of the assignment, you will need to develop a modified version of the scanner developed for Part 1 (e.g., remove the outputs, link to yacc, terminate when errors are found). Keep in mind that **both** versions of the scanner must be handed in and they will be evaluated separately so be sure to keep them separate!

Grading

This assignment is worth 15% of your final grade. The breakdown for this assignment is as follows:

- (40%) Part 1: The lex file that splits up the XML file into tokens. (You are given 5 of the 10 tests that will be run to check the correctness of your code)
- (50%) Part 2: The lex and yacc file that validates the syntax and a bit of the semantics. (You are given 5 of the 10 tests that will be run on your code)
- (10%) Brief documentation included in a README.TXT file. Your readme file should include any assumptions that you made (e.g., how to handle cases that were not covered in the provided specifications) and a disclosure and explanation of the sources you used (e.g., friends, web site, etc) to help you complete this assignment.

Please check the course policy about assignments. Keep in mind that "Exchange of ideas and techniques is encouraged but you must turn in original work and acknowledge the help you receive from others" and that "Each student is given an automatic extension of five (5) calendar days," but some of you have already consumed some of them.

Submission

Use handin (http://cse.unl.edu/~handin) to submit all your documents:

1. Your lex file for part 1 of the assignment (should be labelled assn2-part1.l)
2. Your lex and yacc files for part 2 of the assignment (assn2-part2.l, assn2-part2.y)
3. A readme file on how to compile and run your code.