CS 476 - Programming Language Design - Spring 2007

Homework #1

Due date: February 20th, 2007, 11:59 pm

General description

In this assignment, you will develop a small calculator that executes commands given in natural language. This will help you familiarize with context-free grammars and tools that can be used to build sophisticated applications.

Part 1

Write a context-free grammar that recognizes the language described below. You don't necessarily have to submit the grammar with your solution, but you can submit it if you are not able to complete the rest of the assignment, to receive partial credit.

The language to be represented is a small subset of English, composed by imperative sentences describing commands to be performed. A sentence begins either with the command **tell me** or **yell**, followed by the description of a mathematical expression to be calculated. Examples:

1. Tell me the sum of 2.5 and -3
2. Tell me the difference between 10 and 5.1
3. Yell the square root of 2
4. Yell the product of the square of 5 and the ratio of 10 over 3

As you can infer from example 4, mathematical expressions can be nested arbitrarily. Since the form of the language corresponds to a prefix notation, parentheses are not necessary. The following table contains a list of the valid expressions and their mathematical meaning. X and Y are floating point numbers.

Expression	Meaning
`the sum of X and Y`	$X+Y$
`the difference between X and Y`	$X-Y$
`the product of X and Y`	$X*Y$
`the ratio of X over Y`	X/Y
`the opposite of X`	$-X$
`the inverse of X`	$\frac{1}{X}$
`the square of X`	X^2
`the square root of X`	$\sqrt{X}$

Part 2

Implement a system using the grammar you wrote in the first part of the assignment and the tool **JavaCC**. JavaCC allows you to attach fragments of Java code to the production rules of a context-free grammar. The tool automatically generates a parser that recognizes the language described by the grammar and executes the corresponding code.

JavaCC is freely available at the following website:

```
https://javacc.dev.java.net/
```

The installation of JavaCC is straightforward. To complete this assignment, the main tool (`javacc`) is sufficient, you don't need to use the `jjtree` and `jjdoc` utilities also provided in the package.

Your system will parse sentences provided by the user, perform the appropriate arithmetic calculations, and return the result in a format dependent on the command ("tell me" or "yell"). Examples:

Input	Output
`Tell me the sum of 2.5 and -3`	`The result is -0.5`
`Tell me the difference between 10 and 5.1`	`The result is 4.9`
`Yell the square root of 2`	`THE RESULT IS 1.4142135623730951!!!`
`Yell the product of the square of 5 and the ratio of 10 over 3`	`THE RESULT IS 83.33333333333334!!!`
`qpwoei rieoq fqeij!!!`	`Sorry, I couldn't understand your command.`

For more examples of input/output behavior, you can download and run the reference implementation from the class' website. Note: if you find any mistake or odd behavior in the reference system, please report it to the TA as soon as possible. To run the reference system, use the command:

```
java -jar NLCalculator.jar
```

Tips and additional requirements:

- Read carefully the examples in the directory "SimpleExamples" of the JavaCC package. They are very useful to understand how JavaCC works.
- Make your system case insensitive and whitespace insensitive. This can be done playing with the options and the `SKIP` directive.
- Put your code in a file named `NLCalculator.jj` and call the main class defined into it `NLCalculator`. JavaCC will generate many `.java` files, among them there should be a file called `NLCalculator.java`.
- Make sure your system reads the sentences from the standard input. Each sentence should be delimited by a newline character.
- The system should exit upon the additional commands **quit**, **exit**, or when the end of file is reached.

Part 3

Extend the system such that it can also handle integer numbers expressed in natural language, such as **two hundred thirty five**, **one thousand three hundred fifty**, **seventeen hundred**, and so on. For simplicity, we will limit the interpretation of natural language numbers to integer numbers in the range between **negative nine hundred ninety nine thousand nine hundred ninety nine** (-999999) and **nine hundred ninety nine thousand nine hundred ninety nine** (999999).

Tips:

- Negative numbers can be implemented with a simple extension of the "opposite of" operation.
- Be careful with the `LOOKAHEAD` option and the order in which you specify the rules.

What and how to submit

If you successfully completed all the three parts of the assignment, you should submit two files:

- `NLCalculator.jj`, which contains the code of the extended system, to be compiled with JavaCC. The main class defined in this file should be called `NLCalculator`.
- A plain text `README` file with a description of how you approached and solved the problem.

If you were able to complete part 1 and part 2, but not part 3, you should submit the same two files, adding to the `README` file a detailed description of the problems you encountered in the implementation.

If you could complete part 1 only, the `README` file should contain the description of your grammar, with the same notation used in the book.

You will submit your files using the `turnin` command on the CS server:

```
turnin -c cs476 -p hw1 NLCalculator.jj README
```

You may work on this project by yourself or in a group of two. If you work on this with another member of the class, only submit your project once using `turnin`. Be sure that you clearly state in your `README` file the names and NETIDs of the both members in the group.

Resources

The class' website:
`http://www.cs.uic.edu/~i476/`

The JavaCC home page:
`https://javacc.dev.java.net/`

A tutorial on JavaCC:
`http://www.engr.mun.ca/~theo/JavaCC-Tutorial/`