COMP 520 Compiler Design
Individual Assignment #1

Scanning and Parsing MiniLang

Due: Friday, January 24, 11:59 PM

Overview

In class, we focus primarily on the theoretical side of compilers, with only a brief overview of the tools. The purpose of this first assignment is to get you up to speed with the practical side of scanning and parsing. You may use any toolchain you wish (in class we show flex/bison in C and SableCC in Java) provided that it runs on the SOCS Trottier machines. You may also hand-code a scanner and/or parser if you're interested! Use this time to explore the various options and find a set of tools that you will be comfortable with for your course project.

In this assignment, you will build a compiler for the `MiniLang` language, original defined by Vincent Foley. A formal overview is found at `http://www.cs.mcgill.ca/~cs520/2020/assignments/Assignment1_Specifications.pdf` . Note that this may change as we discover the intricacies of language design - report any issues/challenges you encounter and we will discuss in class!

Question 0: *Name the COMP 520 dragon*

Suggest a name for the dragon on the COMP 520 home page: `http://www.cs.mcgill.ca/~cs520/2020/`. The winning name gets bonus points!

Question 1: *Example Programs* (10 points)

Given the `Minilang` specification, write the following example programs with file extension `.min`

(a) **Syntactically-Correct Example Programs** (5 points)

Write two syntactically correct `MiniLang` programs that perform an *interesting* computation using a variety of language features (no Fibonacci or factorial!).

(b) **Syntactically-Incorrect Example Programs** (5 points)

Write five incorrect `MiniLang` programs which each exhibit a *different* scanning or parsing error. Each program should be minimally sufficient to trigger the error, so think carefully. Include a comment at the start of each file describing the intended issue.

Note that although only 7 test programs are required, you should prepare a more substantial set for debugging. As part of the evaluation, we will test a comprehensive suite for all language features.

Question 2: *Scanner/Parser for `MiniLang`* (20 points)

Implement the scanner and parser for `MiniLang` using your chosen toolchain.

For the first assignment, your compiler must support 3 modes supplied as a command-line argument:

- `scan`: Outputs `OK` if the input is lexically correct, or an appropriate error message
- `tokens`: Outputs the token kinds, one per line, until the end of file. Tokens with associated data (literals, identifiers, etc) should be printed with their respective information
- `parse`: Outputs `OK` if the input is syntactically correct, or an appropriate error message

Normal output (`OK`, tokens) must be sent to `stdout` and your compiler must exit with status code 0. If an error occurs, the message must be displayed on `stderr` and your compiler must exit immediately with status code 1. Error messages should be descriptive (look into error reporting in your toolchain) and formatted "`Error: <description>`". Follow the output specifications **exactly**.

Scripts

To facilitate testing, your assignment must follow the starter code and directory structure provided on the course GitHub repository: `https://github.com/comp520/Assignment-Template`. As part of your submission, provide the following 2 scripts:

- `build.sh`: Builds your compiler using `Make` or similar
- `run.sh`: Takes two arguments (mode and input file) and executes your compiler binary

Comments found in both files provide the exact requirements. A convenience `test.sh` script executes all programs in the `programs` subdirectories and reports any issues found.

Submission

On myCourses, hand in a tar.gz file of the form `firstname_lastname.tar.gz` (all lowercase, use .tar.gz). The structure of files inside the tarball is given below.

A `README` file should include your student ID, as well as any resources that were consulted as part of your submission (other students, websites, repositories, etc.). If no resources were consulted, please state "I worked alone". As this is an individual assignment, all code must represent your own efforts – please check with us if in doubt!

```
firstname_lastname/
    README   (Name, student ID, any special directions for the TAs, resources)
    DragonName.txt (Answer to Question 0)
    programs/
        1-scan+parse/
            valid/   (Two correct programs)
            invalid/ (Five syntactically incorrect programs)
    src/     (Source code and build files)
    build.sh (Updated build script)
    run.sh   (Updated run script)
```