Project 1

In this project, students will create a lexer for the Geaux language using ANTLR4 (https://www.antlr.org/)

Geaux

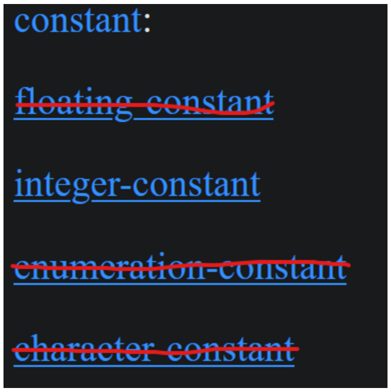
Geaux is a stripped down C. We will be using a modified version of the C specification, which can be found at: https://port70.net/~nsz/c/c89/c89-draft.html#3.1

We will make several deviations. First, these are the keywords, operators, and punctuators we will be using:

```
Geaux keywords: var fun while const string void return if
else break int typedef struct union
Geaux operators: < && || * + ~ = . ->
Geuax punctuators: { } , ( ) & | ! ~ ; : [ ]
```

We will not be implementing preprocessing-tokens. Identifier tokens will be implemented using the C specification at: port70.net/~nsz/c/c89/c89-draft.html#identifier

Students will not be responsible for implementing floating-constant, enumeration-constant, or character-constant.



constant:

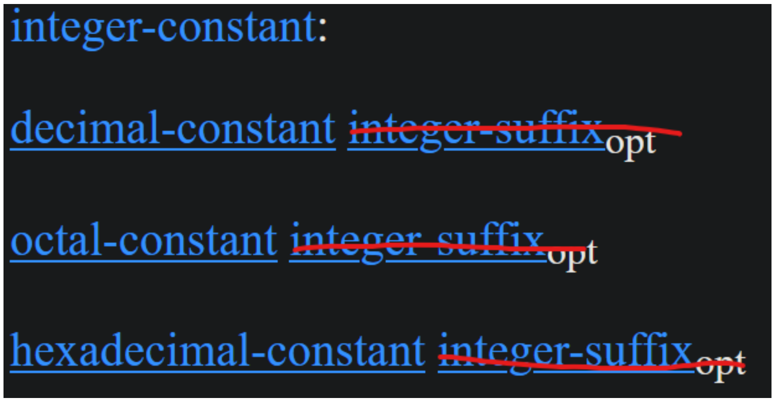
~~floating-constant~~

integer-constant

~~enumeration-constant~~

~~character-constant~~

In integer-constant, students do not have to implement the following crossed out items:



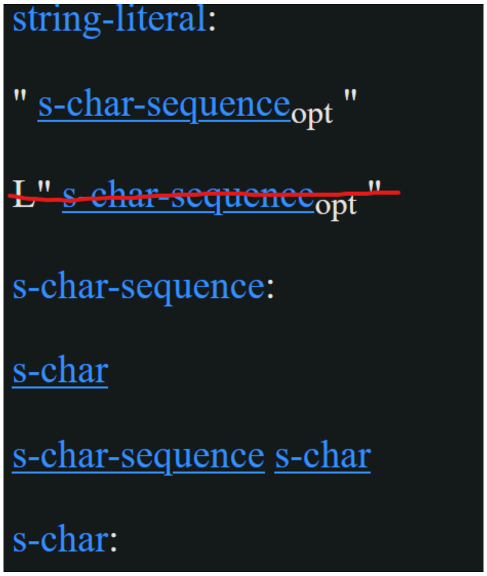
integer-constant:

decimal-constant ~~integer-suffix~~$_{opt}$

octal-constant ~~integer suffix~~$_{opt}$

hexadecimal-constant ~~integer-suffix~~$_{opt}$

Geaux will not have an integer suffix, but the rest will be unchanged from C. Lastly, string-literal will have a slight change. Students will not be responsible for the following crossed out item:



string-literal:

" s-char-sequence$_{opt}$ "

~~L" s-char-sequence$_{opt}$ "~~

s-char-sequence:

s-char

s-char-sequence s-char

s-char:

Building & Running Prog1

A script to run the project has been provided (run.sh). Additionally, a script to run the solution has been provided as well (run_solution.sh). Inside of these scripts include the steps to build the project. All of the .class files generated from the build will be placed into the bin folder.

Testing Prog1

For grading, we have a test case generator that generates random tests cases. We have used it to generate 100 test cases for you to use. These can be found in the tests folder. We have a test.sh script that will run your version, and our version and compare the results.

Grading

We will generate 10 random test cases, and then inspect the output of your program against those test cases. The output should match our output. For superficial differences, no points will be lost, but for meaningful differences, points will be lost. Try to match my output exactly, that’s the easiest way to get a 100.

Academic Honesty

You are allowed to use AI, online resources, ANTLR documentation, etc. You are NOT allowed to use a fellow student’s implementation. Any group found in violation of this will face the penalties specified in the syllabus.