CS330 – Midterm Project: Baby Compiler/Syntax checker

Due: March 5, 2016

Goal: Write a program, in Perl, that evaluates the correctness of the syntax of a simple Java program.

For C level credit:

- Develop a compiler for a simple driver only, that is, a class that contains only the method main.
- Read in the entire program (sample development code found in CLevel.java) into a single scalar.
- Use pattern matching to ensure that the program conforms to the following Java language rules;
    a. The class definition, declaration of the static main, etc. conform to Java rules for the driver.
    b. Rules for declaring identifiers conform to Java language rules. You do not have to handle object or array declarations, just primitives.
    c. All reserved words are recognized as reserved words and treated accordingly.
    d. Rules for assignments and arithmetic statements are passed/ recognized if they conform to Java rules. You don't need to concatenate Strings.
    e. Rule for simple output: System.out.println("a single String"); or System.out.println(aVariable);
    f. Any other structures not explicitly mentioned here (e.g. if/ switch) can be ignored at this time.

For B-level credit: Do everything listed in the C-level. In addition,

- Include if, if/else, for and while statements, which will be recognized as correct if they conform to their Java rule forms. Assume that branching statements must have {}'s even if they contain a single statement (this actually makes the problem easier).
- In order to recognize this expanded inventory, you will have to recognize correct Boolean operations.
- Rules for String concatenation ("The answer is " + blacktail);
- Sample code that must be correctly compiled is listed in BLevel.java

For A-level credit:

- Add Input (Scanner declaration and use);
- Recognize object declarations (you don't have to ascertain if the declaration matches the constructor, just if it has the right general form).
- Recognize a code block that is a class definition, including attributes, constructor(s) and methods.
- Sample code that must be correctly compiled is listed in Alevel.java

A few (hopefully) helpful hints:

- Try to maintain readable bottom-up pattern matching throughout your program by substituting lower level constituents for representative symbols (e.g. replace all valid identifiers with **I,** as in **$program =~ s/[a-zA-Z][a-zA-Z0-9_]*/I/g;**)
- Make sure you comment each line to explain what it's replacing
    - Printing the program after each line (with an explanation of what was supposed to happen) will help too
- Start with the C level project, and build up from there
    - In this project, building B work on a C project (and then A work on B) may require adding steps (pattern matching and/or replacement statements) BEFORE others that made up the strategy used in the prior project. If you want, you can read in the test program more than once, or save every line for later use.
- Remember that in Java, the convention is to put everything on a different line and to indent, but it will compile without it. You might find it helpful to get rid of the white space in the input file, or at least substitute a long stretches of white space with single spaces
- Speaking of whitespace, to simplify the project you can assume that there will always be a single space separating
- The order that you do the replacements is really important. Really, really important. Try to draw out a parse tree, if you find that to be helpful.
- You can assume that names will adhere to Java conventions: variables will always start with a lower case letter, and class names will always start with upper case letters.

A note on grading:

Your final grade will be based on what level assignment you turn in, but I will be testing your syntax checkers on code that is slightly different from what I gave out. Likely changes include: switching the order of the code in main, changing variable names, and adding in more expressions in general. You are encouraged to make changes to your own test code to make sure that your syntax checkers can handle these changes.