**Programming Assignment#3**

Your goal is to rewrite my Bison-based While language compiler (lexical analyzer/parser/intermediate code generator) to accommodate changes to the language. Our new language will be called the While++ language.

**THE CHANGES FROM WHILE TO WHILE++**

Programs will no longer start with a list of variables. Rather, they must start with the keyword “class” which is followed by a class name. The class encompasses its definition in braces. The definition contains public (exposed) variable declarations, private (unexposed) variable declarations, and a body of code. For now, all exposed declarations start with the keyword “int” and unexposed ones start with “private int”. The body of code can reference any variables declared in its class, whether exposed or unexposed. Later, when we define the project, we will expand the definition of While++ to contain multiple classes and discuss how access to the variables of other classes will be included, but that is not a goal of this assignment. Thus, we have just one class.

Unlike the original While language, the assignment operator is now the single symbol “=” and the equality operator is “==” not “=”. Oh, and unlike the original While language, test expressions used in “while…do”, “do…while” and “if” must be surrounded by parentheses. Additionally, underscores may be embedded in identifiers, but they are ignored as in your prior assignment; all identifiers are truncated to a max of eight characters; and case is no longer relevant in either keywords or identifiers. Finally, the program body is a block of statements, so it must start with a left brace and end with a right brace. Of course, blocks can be embedded within this block, but all declarations must appear prior to (in the exposed section) or at the start of the outer block.

class Sample {

int a, b[26];

private int index;

A = 14;

while (index<26) do {

b[index] = a\*index; index = index+1; }

}

Here are a few other changes:

The keyword “then” is optional in an “if” statement. If omitted, it is assumed. Similarly, the “do” is optional on a “while…do”, but not on a “do…while”.

A simple “for” statement must be implemented. That statement has two forms  
for up (index\_variable=expression; test) statement   
for down (index\_variable=expression; test) statement

The “for up” increments its index variable by one at the end of the loop; the “for down” decrements its index variable by one at the end of the loop. Note the word “up” can be omitted – thus, for (…) is equivalent to for up (…). This shows the “for” statement.

class Sample {

int a, b[26];

private int index;

a = 14;

for (index = 0; index<26) b[index] = a\*index;

}

One more thing, in the language I implemented, semicolon is a separator. In While++, semicolons are terminators for all primitive statements (including declarations, but not blocks).

**What you must do**

Rewrite my while.l and while.y to handle the changes syntactically and with appropriate intermediate code being generated. Assume execution starts at the first statement of the one class, so that’s not at all different from the code I generated. You will have to generate code for an if-statement, but that’s the extent of new code; the rest of the changes are syntactic.

This assignment is due on November 7 at 11:59PM.