Assignment 3

CSI 4336

Due September 18, 2018

Submitting your assignment

All written portions of the assignment should be prepared in LaTeX.

Submit this assignment by the due date in two ways: by email (before class) and printed (at the beginning of class). Don't put any code in the printed copy. Proofread your document for style before submitting it.

Send the email to `hamerly@cs.baylor.edu` with the subject "CSI 4336 assignment X" (where X is the assignment number). The email should have one attachment (plain text, .zip, or .tar.gz format) containing:

- the .tex document you wrote named "lastname.tex" (where 'lastname' is your last name),
- a compiled .pdf from the .tex document named "lastname.pdf" (where 'lastname' is your last name),
- any additional files used in your LaTeX document, named "lastname_fig1.pdf" (or similar), and
- all source code used for any programs.

1 Textbook exercises and problems (10 points each, 20 points total)

- Do exercise 2.2 from your textbook. You may assume the result from example 2.36, and you may also assume that the class of CFLs is closed under union.
- Do problem 2.30 (part a) from your textbook.

2 CYK parsing algorithm (20 points)

Do the problem 'Context-Free Grammar Parsing' on Kattis. Read on for a description of the central algorithm.

The CYK (Cocke-Younger-Kasami) algorithm is a bottom-up parser which uses dynamic programming to fill in a table T. Let the string we are parsing be $s = s_0 s_1 \dots s_{n-1}$, so the length of the string is n. The table is of size $n \times n$, and the table is indexed from $T[0,0]$ to $T[n-1, n-1]$ (though only half of the table is actually used). Each entry $T[i,j]$ in the table contains a set of grammar variables which could have generated the part of the string $s_j s_{j+1} \dots s_{j+i}$. The table is filled from bottom to top, and if $T[n-1, 0]$ contains the start variable, then s is in the language of the grammar (otherwise, not).

The bottom row of the table is initialized with the variables which could have produced each terminal:

$$T[0,j] = \{X | \text{ the production } X \rightarrow s_j \text{ exists}\}, 0 \leq j < n$$

Each entry in the second row ($T[1,j], 0 \leq j < n-1$) will be filled with the variables that could have produced any of $T[0,j] \times T[0,j+1]$. Each entry in the third row ($T[2,j], 0 \leq j < n-2$) will be filled with the variables that could have produced $T[0,j] \times T[1,j+1]$, or $T[1,j] \times T[0,j+2]$. Note that on this row we need to consider more than one way to produce $T[2,j]$. On the next row there are more possibilities to consider. This continues until we reach the top of the table, at $T[n-1,0]$.

Let's consider an example, with the grammar given above, parsing the string 'aacbb'. The final table will look like this:

$T =$

i\j	0	1	2	3	4
0	A	A	A,B	B	B
1	A,S	A,S	A	$\emptyset$	
2	A,S	A,S	A		
3	A,S	A,S			
4	A,S				

and this string will be accepted because S appears in the entry $T[4,0]$.

While your textbook does not discuss the CYK algorithm, there are many references and textbooks on programming languages, parsing, and compilers which do. Using the correct data structures to represent the table and the grammar helps simplify your code. My solution to this problem is less than 80 lines of well-formatted C++.