COSC 4316

Take Home Exam

Due Tuesday, April 17, 11:59 p.m.

This exam is a take-home examination. The exam is open book, but **no collaboration is allowed.** Specifically, you may not communicate with any person about any aspect of the exam until after the hand-in deadline, even if you have already handed in your exam.

Attach this cover sheet and the problem sheet to your exam solutions as you turn it in your work.

Name _____
(printed)

I certify that I have not collaborated with anyone else in the solution of these problems. I understand that the penalty for any academic dishonesty will be severe, up to, and including failure for this course.

Signature_____

1. Construct a CFG for parsing regular expressions over the binary alphabet $\Sigma = \{a, b\}$. (No, the language for generating regular expressions is not regular, but is context-free. *Hint: think about our grammar for infix expressions*.) Draw a parse tree for the r.e.:

   `a(ab)*(a|b)`

2. C and Java allows loop exits using the "break" and "continue" statements. A "break" statement leaves the current loop entirely, and a "continue" statement jumps to the end of the current loop. A break statement can only appear inside a loop or a switch statement, and a continue can only appear inside a loop. If we add "break" and "continue" to ZinC, what changes would have to be made to the semantic analyzer to ensure that these statements only appear within loops? What changes would be needed for the abstract stack machine for which you wrote the assembler in order to implement "break" and "continue"?

3. Use the action/goto table for the grammar below (where the productions are as numbered) to parse the incorrect input string:

   `( 2 () ) 5 )6 )`

   When the syntax error is detected, a) what does the stack look like, b) what was the last action and c) what is the lookahead?

   1: R → S
   2: S → (L) L
   3: S→ ε
   4: L → int L
   5: L → S

   `int` token matches any integer (i.e. 2,5,6 in input)

State	(	)	int	$	S	L
0	s2	r3		r3	1	
1				acc		
2	s2	r3	s4	r3	5	3
3		s6				
4	s2	r3	s4	r3	5	7
5		r5		r5		
6	s2	r3	s4	r3	5	8
7		r4		r4		
8		r2		r2		

4. The grammar in the previous problem (with start symbol R and non-terminal symbols R, S and L) generates 'lists' of integers and lists. For example,

   `( 1 ( 3 7 6 ) 2 ( 4 ) )` can

   be generated using the grammar.

   - Write rules for each production that compute the sum of the items in the list. For the example above, you would want '23' printed out at the end. Clearly state what attribute(s) you intend to use; do not use any global variables. You may use either **`yacc`** notation or the attribute grammar notation used on the slides. In general, I don't care about the correctness of your syntax but your meaning should be clear. You can assume the int token has a val attribute that holds the integer value of the lexeme.
   - Draw the parse tree for the input above and show how your attributes are used for the computation.

5. Rewrite the following grammar to be LL(1):

   A → Abb | Abc | cc

6. How Much Wood ... ?
   Consider the following grammar over the tokens {if, wood, could, chuck, a, howmuch}:

   S → Q OptCond | Stmt OptCond
   OptCond → **`if`** Stmt | ε
   Stmt → N V OptObj
   V → **`wood`** V | **`could`** V | **`chuck`**
   OptObj → N | ε
   N → **`a woodchuck`** | **`wood`**
   Q → **`howmuch wood wood`** Stmt | **`could`** Stmt

   This grammar allows us to generate sentences such as "**`howmuch wood wood a woodchuck chuck if a woodchuck could chuck wood`**."

   a. The professor verified that this grammar is unambiguous using a piece of software you should be familiar with. In one sentence, how do you think he did this?

   b. In one sentence, what are the drawbacks to the method proposed above?