Final Exam Due Sat 12/15 @ 11:59PM

CMPS 4500 FALL 2019

Name: _____ _____/100

****

QUESTION #1 (20 pts)

****

Design a deterministic finite automaton for the set of all strings that start with 'a' followed by nothing or any combination of 'a' or 'b' and end with 'ab' over the alphabet X = {a,b}. The shortest string in this language is 'aab'. Your DFA should match the longest pattern; i.e, it should not stop at the first 'ab' in string aabbbbab but read all the way to the end of input.

a) Provide the regex for this language and the transition function t as a diagram or a digraph. Recall that a DFA is a 5-tuple <S, X, s0, F, t> where

S = finite set of states

X = finite set of input alphabet

s0= start state

F = set of final states, where F is a subset of S

t = transition function

In the next question use the following notation:

[0,aabab] -> [1,aabab] // meaning transition to state 1 on input a

b) Show computation for 'aabab'

c) Show computation for 'aaaa'

d) Show computation for 'aaba'

****

QUESTION #2 (20 pts)

****

Given this BNF Expression Grammar G, where n is a terminal in {0..9}

E -> E + n | n

a) remove left recursion without modifying the meaning of the grammar

For the new grammar

b) show the LL(1) parsing stack for expression 3 + 4 + 5.

c) show the First and Follow sets

****

QUESTION #3 (20 pts)

****

Let G be the grammar defined by

X = { SEMI , ID} // SEMI and ID are terminals

N = { id_seq } // id_seq is the only nonterminal

S = id_seq

P = { id_seq -> id_seq SEMI ID | ID }

a) augment the grammar

b) construct the SLR sets of items for the augmented grammar

c) compute the unique closure set for the SLR sets of items; i.e., the
canonical collection of items for the augmented grammar.
d) compute the GOTO function for the canonical collections of items.
e) construct the parsing table for the grammar
f) show the actions of your parsing table on input
a;b
note:'a' and 'b' are ID tokens and ';' is SEMI token

b) construct the SLR sets of items for the augmented grammar
c) compute the unique closure set for the SLR sets of items; i.e., the
canonical collection of items for the augmented grammar.
d) compute the GOTO function for the canonical collections of items.
e) construct the parsing table for the grammar
f) show the actions of your parsing table on input
a;b
note:'a' and 'b' are ID tokens and ';' is SEMI token

****

QUESTION #4 (20 pts)

****

//Note: you can implement this problem using yacc/lex if you wish.

The following BNF grammar defines an expression language over +, *, unary -,
and identifiers a and b.

<expr> ::= <term> | <expr> + <term>
<term> ::= <id> | <term> * <id>
<id> ::= a | b (<expr>) | - <expr>

Design and implement an SDT that translates expressions into nested function calls. For example, given expression

(a+b) * -(b+c)/a

your SDT will generate this code

times(plus(a,b),times(negative(plus(b,c)),a)).

Hint: you can construct this entirely with synthesized attributes using a concatenation operator in the same way that you generate 3-address code.

****

QUESTION #5 (20 pts)

****

1) (5 pts) What functionality is in the middle-end and back-end compiler phases? These two phases are often combined into one phase called the back-end.

2) (5 pts) In what compiler phases is the symbol table created and in what phases is the symbol table used?

3) (5 pts) What is the difference between a lexeme and a token? Give an example of each in C.

4) (5 pts) Categorize and count the lexemes in this valid C code:

```
int i = '3';
*nptr = pow(2,i) + j++;    /* stuff stuff */
```