CPSC 323 FINAL STUDY GUIDE

Given the following productions, and the string w = ‘abab’. Is this grammar ambiguous and if so, explain ambiguity and show why? (30)

1) E –> aEbE
2) E –> bEaE
3) E –> cdE
4) E –> ε

Identify and remove all left recursions in the following productions. (30)

1. E –> EaE
2. E–> F
3. F–> mE
4. F–> Tde
5. F –> Tfg
6. T –> id

Find the First and Follow sets for each non-terminal symbol (30)

D –> ONE

O –> xy | b | ε

N –> Ez | cd | ε

E –> m | fg | ε

Given the following productions, construct the parsing table for table driven predictive parser - it is a top-down parser. (30)

1) S –> S & C

2) S –> S @ C

3) S –> C

4) C –> x

Given the following production rules, write a recursive descent function that returns a boolean value for the productions Q. Write the function using syntactically correct C, C++, C#, python (for 30 points) or pseudo code (20 points). Use any of the pre-existing functions lexer(), getNextChar(), currentChar(), first(), follow(), token(), backup(), error() or match() only if needed.

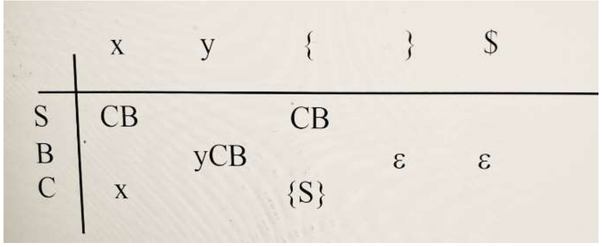
1) E –> TQ

2) Q –> #TQ

3) Q –> ε

4) T –> id

Given the following predictive parsing table, parse the string “ *{xyx}* ”
(30 points)



	x	y	{	}	$
S	CB		CB		
B		yCB		ε	ε
C	x		{S}		

Use any of the following: **Stack input Production/Action**

For the following NFA state transition table function:

	a	b	epsilon
0	{}	{2}	{1}
1	{0,4}	{}	{ }
2	{ }	{4}	{ }
3	{4}	{ }	{ }
4	{ }	{ }	{3}

q0=0 and F={4}

Define the e-closures (5) and convert it into a DFA table using the subset method (15 points)

Use Thompson’s construction method to convert the following RE= a* (a | b) into a NFA diagram (20)

Convert the following Regular Expression into an NFA diagram (10) and into a DFSM table (10). For Sigma={l,d,o} , you can use ‘l’ for letters, ‘d’ for digits and ‘o’ as other inputs for any other symbols. Label the starting state and final state. RE = l (l | d)*o

Based on the book, write the code for a DFSM() function that can iterates through a state transition table[1..nstates, 1..nInputs] and determine if an input string(w) is accepted or not. Given that w is a string(1D-array of chars as a parameter) and table is a 2D-array(of numerical values and has already been pre-defined). And the function char_to_col() can convert any character to an integer value. You can also use any of the pre-existing functions getNextToken(), currentToken(), getNextChar(), currentChar(), backup(), error() or match() if needed or not. Using pseudo-code to implement the function (maximum of 15 points) Using syntactically and grammatically correct c/c++ or python (maximum 20 points)