COVER PAGE

CS323 Programming Assignments

Fill out all entries 1 - 6. If not, there will be deductions!

Peer Review (Check one)

1. Names [1. Lambert Liu], (ThumbUP [✓] or ThumbDown [ ])

[2. Kun Fang], (ThumbUP [✓] or ThumbDown [ ])

[**if 3**.], (ThumbUP [ ] or ThumbDown [ ])

2. Assignment Number [2]

3. Turn-In Dates: **Final Iteration with Documentation** [2019/11/16]

4. Executable FileName [Run.exe]
(**A file that can be executed without compilation by the instructor**)

5. LabRoom [CS101]
(Execute your program in a lab in the CS building before submission)

6. Operating System/Language [C/C++]

To be filled out by the Instructor:

GRADE:

COMMENTS:

Documentation

1. Problem statement.

syntax analyzer:

1. Rewrite the grammar provided to remove any left recursion
 (Also, use left factorization if necessary)
2. Use the lexer() generated in the assignment 1 to get the tokens
3. The parser should print to an output file the tokens, lexemes and
 the production rules used;
 That is, first, write the token and lexeme found
 Then, print out all productions rules used for analyzing this token
 Note: - a simple way to do it is to have a “print statement” at the beginning of each function that will print
 the production rule.
 - It would be a good idea to have a “switch” with the “print statement” so that you can turn it on or off.
4. Error handling: if a syntax error occurs, your parser should generate a meaningful error message, such as
token, lexeme, line number, and error type etc.
Then, your program may exit or you may continue for further analysis.
The bottom line is that your program must be able to parse the entire program if it is syntactically correct.
5. Turn in your assignment according to the specifications given in the project outline

2. How to use your program:

Double click Run.exe

3. Design of your program:

---------------------designed production rule---------

R = {

S->id=E

E->E+T|E-T|T

T->T*F|T/F|F

F->(E)|id

}

---------------------after removing left recursions-----

R = {

S->id=E 1)

E->TQ 2)

Q->+TQ|-TQ|epsilon 3)4)5)

T->FR 6)

R->*FR|/FR|epsilon 7)8)9)

F->(E)|id 10)11)

}

---------------------------G=(N,T,S,R)--------------------------

N = {S,E,T,Q,R,F}

T = {+,-,*,/,=,(,),;,id,epsilon}

S = S

----------------------FIRST-----------

FIRST(S) = {id}

FIRST(E) = {(,id}

FIRST(T) = {(,id}

FIRST(Q) = {+,-,epsilon}

FIRST(R) = {*,/,epsilon}

FIRST(F) = {(,id}

--------------------FOLLOW----------------

----note: ';' = '$' means empty, or end of string

FOLLOW(S) = {$}

FOLLOW(E) = {),$}

FOLLOW(T) = {+,-,), $}

FOLLOW(Q) = {),$}

FOLLOW(R) = {+,-,), $}

FOLLOW(F) = {+,-,*,/,),$}

-----------------------table---------------------

	id	+	-	*	/	=	(	)	$
S	1	@	@	@	@	@	@	@	@
E	2	@	@	@	@	@	2	@	@
Q	@	3	4	@	@	@	@	5	5
T	6	@	@	@	@	@	6	@	@
R	@	9	9	7	8	@	@	9	9
F	11	@	@	@	@	@	10	@	@

4. Any Limitation:

   A single sentence can not greater than 1000 characters

5. Any Shortcoming:

   A) Text file format must be correct: every word or operator should separate by space
   B) Only support format: Id = Expression + Expression ;
   C) End of the sentence must end with ;