University of Central Punjab

Faculty of Information Technology

Compiler Construction

Project Phase # 2

Submission Before: 11:55PM - 23-12-2019
(Late will be penalty of deduction of 2 absolute marks per day)

TINY-C++:

This is a subset of C++ language. Description of the language as follow:

		Detail	Example
1	Identifiers	(_|L)(L|_|D)*(D|_)	_rate2, _rate_, rat1e2 …etc
2	Numbers	[+-]?(D+)(\.D+)?, and exponent numbers.	3.43433E+13, 3.33333E-25 +34.5, -34.5, 34.5, 34
3	Operators	<,>,!=, <>, :=, ==, *, +, /,-, >>,<<, ++, +=, &&, ||, <=, >=, %, “ ”	
4	Punctuations	[,{,(,),},]	
5	Keyword	asm else new this auto enum operator throw bool explicit private true break export protected try case extern public typedef catch false register typeid char float typename class for return union const friend short unsigned goto signed using continue if sizeof virtual default inline static void delete int volatile do long struct double mutable switch while namespace template Input->, output<-	

Grammar

Right down grammar for C-Like syntax

```
Function     →     Type identifier ( ArgList ) CompoundStmt
ArgList      →     Arg   | ArgList ,Arg
Arg          →     Type identifier
Declaration  →     Type IdentList ;
Type         →     int   | float
IdentList    →     identifier ,IdentList   |     identifier
Stmt         →     ForStmt     | WhileStmt | Expr ;     | IfStmt
                   | CompoundStmt    | Declaration      | ;
ForStmt      →     for ( Expr ; OptExpr ; OptExpr ) Stmt
OptExpr      →     Expr  | ε
WhileStmt    →     while ( Expr ) Stmt
IfStmt→     if ( Expr ) StmtElsePart
ElsePart     →     else Stmt | ε
CompoundStmt→{ StmtList }
StmtList     →     StmtListStmt | ε
Expr         →     identifier :=Expr | Rvalue
Rvalue→     Rvalue Compare Mag| Mag
Compare      →     == | < | > | <= | >= | != | <>
Mag          →     Mag + Term
                   | Mag - Term
                   | Term
Term         →     Term * Factor
                   | Term / Factor
                   | Factor
Factor→     ( Expr )
                   | identifier
                   | number
```

Assignment Description:

For this assignment,

1. Implement the Sub-Grammar only.
2. You have to write a **Parser**for above Grammar.
3. **Parser** will get **Token** from scanner and built a parse tree.
4. Parser will built the parse tree using Predictive Parser (LL(1)) grammar.
5. **Panic Mode** approach will be implemented to output the syntax error.
6. This assignment includes following parts:

	PARTS	**Output**	**Marks**
1	Convert grammar to LL(1) grammar.		15

2	Implement Parser using i) LL1 Parsing Table	Source Code Files	60
3	Generate Parse Tree		25
	Total		100
	Absolute		**7**

Rules:

1. This is an individual assignment. Each student has to submit his/her assignment work.
2. Group discussion is allowed but don't share code and other part of assignment with other student.
3. Plagiarism is not tolerable in any of its form. Minimum penalty would be an '0' marks in the project module.

Tools:

Language (For Development): C++

Note: Student cannot use built-in data structure. Student can use his own data structure Hash Table, Linked List which he/she developed in data structure course. In this case student should know about the data structure.

Evaluating Criteria:

1. Source code should reflect the detail given in documents (other parts).
2. A text file with valid source code will be input of the scanner and Token file will be output of the scanner tool.
3. A text file will show the productions in separate lines used in building the parse tree.
4. A text file show the errors generated from both scanner and parser.