Syntactic Analyzer

Compiler Design (COMP 442/6421)
Assignment 2

Name: Garvit Kataria
Student Id: 40155647

Transformed grammar into LL(1)

START -> PROG .

ADDOP -> plus .
ADDOP -> minus .
ADDOP -> or .

APARAMS -> EXPR REPTAPARAMS1 .
APARAMS -> .

APARAMSTAIL -> comma EXPR .

ARITHEXPR -> TERM RIGHTRECARITHEXPR .

ARRAYSIZE -> lsqbr ARRAYDASH .
ARRAYDASH -> intnum rsqbr | rsqbr .

ASSIGNOP -> equal .

ASSIGNSTAT -> VARIABLE ASSIGNOP EXPR .

EXPR -> ARITHEXPR EXPRDASH .
EXPRDASH -> RELOP ARITHEXPR | .

FACTOR -> REPTVARIABLE0 id FACTORDASH .
FACTOR -> intlit .
FACTOR -> floatlit .
FACTOR -> lpar ARITHEXPR rpar .
FACTOR -> not FACTOR .
FACTOR -> SIGN FACTOR .

FACTORDASH -> REPTVARIABLE2 | lpar APARAMS rpar .

FPARAMS -> id colon TYPE REPTFPARAMS3 REPTFPARAMS4 .
FPARAMS -> .

FPARAMSTAIL -> comma id colon TYPE REPTFPARAMSTAIL4 .

FUNCBODY -> lcurbr REPTFUNCBODY1 rcurbr .

FUNCDECL -> FUNCHEAD semi .

FUNCDEF -> FUNCHEAD FUNCBODY .

FUNCHEAD -> func id lpar FPARAMS rpar arrow RETURNTYPE .

FUNCTIONCALL -> REPTVARIABLE0 id lpar APARAMS rpar .

IDNEST -> id IDNESTDASH .

IDNESTDASH -> REPTIDNEST1 dot | lpar APARAMS rpar dot .

IMPLDEF -> impl id lcurbr REPTIMPLDEF3 rcurbr .

INDICE -> lsqbr ARITHEXPR rsqbr .

MEMBERDECL -> FUNCDECL .
MEMBERDECL -> VARDECL .

MULTOP -> mult .
MULTOP -> div .
MULTOP -> and .

OPTSTRUCTDECL2 -> inherits id REPTOPTSTRUCTDECL22 .
OPTSTRUCTDECL2 -> .

PROG -> REPTPROG0 .

RELEXPR -> ARITHEXPR RELOP ARITHEXPR .

RELOP -> eq .
RELOP -> neq .
RELOP -> lt .
RELOP -> gt .
RELOP -> leq .
RELOP -> geq .

REPTAPARAMS1 -> APARAMSTAIL REPTAPARAMS1 .
REPTAPARAMS1 -> .

REPTFPARAMS3 -> ARRAYSIZE REPTFPARAMS3 .
REPTFPARAMS3 -> .

REPTFPARAMS4 -> FPARAMSTAIL REPTFPARAMS4 .
REPTFPARAMS4 -> .

REPTFPARAMSTAIL4 -> ARRAYSIZE REPTFPARAMSTAIL4 .
REPTFPARAMSTAIL4 -> .

REPTFUNCBODY1 -> VARDECLORSTAT REPTFUNCBODY1 .
REPTFUNCBODY1 -> .

REPTIDNEST1 -> INDICE REPTIDNEST1 .
REPTIDNEST1 -> .

REPTIMPLDEF3 -> FUNCDEF REPTIMPLDEF3 .
REPTIMPLDEF3 -> .

REPTOPTSTRUCTDECL22 -> comma id REPTOPTSTRUCTDECL22 .
REPTOPTSTRUCTDECL22 -> .

REPTPROG0 -> STRUCTORIMPLORFUNC REPTPROG0 .
REPTPROG0 -> .

REPTSTATBLOCK1 -> STATEMENT REPTSTATBLOCK1 .
REPTSTATBLOCK1 -> .

REPTSTRUCTDECL4 -> VISIBILITY MEMBERDECL REPTSTRUCTDECL4 .
REPTSTRUCTDECL4 -> .

REPTVARDECL4 -> ARRAYSIZE REPTVARDECL4 .
REPTVARDECL4 -> .

REPTVARIABLE2 -> INDICE REPTVARIABLE2 .
REPTVARIABLE2 -> .

RETURNTYPE -> TYPE .
RETURNTYPE -> void .

RIGHTRECARITHEXPR -> .
RIGHTRECARITHEXPR -> ADDOP TERM RIGHTRECARITHEXPR .

RIGHTRECTERM -> .
RIGHTRECTERM -> MULTOP FACTOR RIGHTRECTERM .

SIGN -> plus .
SIGN -> minus .

STATBLOCK -> lcurbr REPTSTATBLOCK1 rcurbr .
STATBLOCK -> STATEMENT .
STATBLOCK -> .

STATEMENT -> if lpar RELEXPR rpar then STATBLOCK else STATBLOCK semi .
STATEMENT -> while lpar RELEXPR rpar STATBLOCK semi .
STATEMENT -> read lpar VARIABLE rpar semi .
STATEMENT -> write lpar EXPR rpar semi .
STATEMENT -> return lpar EXPR rpar semi .
STATEMENT -> REPTVARIABLE0 id STATEMENTDASH .

STATEMENTDASH -> lpar APARAMS rpar semi | REPTVARIABLE2 ASSIGNOP EXPR semi .

STRUCTDECL -> struct id OPTSTRUCTDECL2 lcurbr REPTSTRUCTDECL4 rcurbr semi .

STRUCTORIMPLORFUNC -> STRUCTDECL .
STRUCTORIMPLORFUNC -> IMPLDEF .
STRUCTORIMPLORFUNC -> FUNCDEF .

TERM -> FACTOR RIGHTRECTERM .

TYPE -> integer .
TYPE -> float .
TYPE -> id .

VARDECL -> let id colon TYPE REPTVARDECL4 semi .

VARDECLORSTAT -> VARDECL .
VARDECLORSTAT -> STATEMENT .

VARIABLE -> REPTVARIABLE0 id REPTVARIABLE2 .

REPTVARIABLE0 -> IDNEST REPTVARIABLE0 .
REPTVARIABLE0 -> .

VISIBILITY -> public .
VISIBILITY -> private .

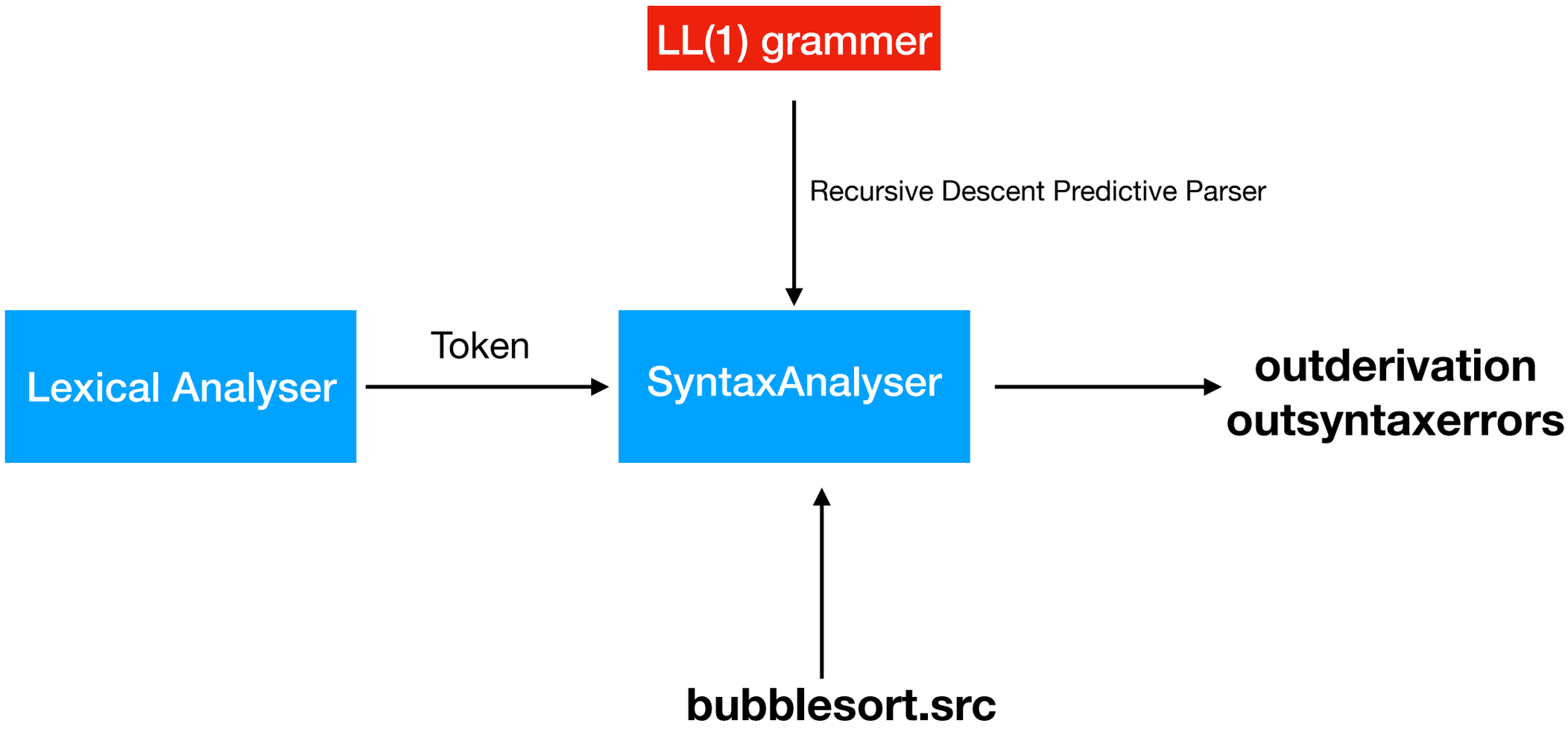
FIRST and FOLLOW sets

nonterminal	first set	follow set	nullable	endable
START	struct impl func	∅	yes	yes
ARRAYDASH	intnum rsqbr	semi lsqbr rpar comma	no	no
ASSIGNSTAT	id	∅	no	no
EXPRDASH	eq neq lt gt leq geq	comma rpar semi	yes	no
FACTORDASH	lpar lsqbr	mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar semi	yes	no
FUNCBODY	lcurbr	struct impl func rcurbr	no	yes
FUNCHEAD	func	semi lcurbr	no	no
FPARAMS	id	rpar	yes	no
FUNCTIONCALL	id	∅	no	no
IDNESTDASH	dot lpar lsqbr	id	no	no
FUNCDECL	func	rcurbr public private	no	no
PROG	struct impl func	∅	yes	yes
ARITHEXPR	id intlit floatlit lpar not plus minus	rsqbr eq neq lt gt leq geq comma rpar semi	no	no
RELOP	eq neq lt gt leq geq	id intlit floatlit lpar not plus minus	no	no
APARAMSTAIL	comma	comma rpar	no	no
REPTAPARAMS1	comma	rpar	yes	no
REPTFPARAMS3	lsqbr	rpar comma	yes	no
FPARAMSTAIL	comma	comma rpar	no	no
REPTFPARAMS4	comma	rpar	yes	no
REPTFPARAMSTAIL4	lsqbr	comma rpar	yes	no
REPTFUNCBODY1	let if while read write return id	rcurbr	yes	no
REPTIDNEST1	lsqbr	dot	yes	no
REPTIMPLDEF3	func	rcurbr	yes	no
REPTOPTSTRUCTDECL22	comma	lcurbr	yes	no
REPTPROG0	struct impl func	∅	yes	yes
MEMBERDECL	let func	rcurbr public private	no	no
ARRAYSIZE	lsqbr	semi lsqbr rpar comma	no	no
INDICE	lsqbr	mult div and lsqbr dot rsqbr eq neq lt gt leq geq equal plus minus or comma rpar semi	no	no
RETURNTYPE	void integer float id	semi lcurbr	no	no
ADDOP	plus minus or	id intlit floatlit lpar not plus minus	no	no
RIGHTRECARITHEXPR	plus minus or	rsqbr eq neq lt gt leq geq comma rpar semi	yes	no
MULTOP	mult div and	id intlit floatlit lpar not plus minus	no	no
SIGN	plus minus	id intlit floatlit lpar not plus minus	no	no

FIRST and FOLLOW sets

REPTSTATBLOCK1	if while read write return id	rcurbr	yes	no
RELEXPR	id intlit floatlit lpar not plus minus	rpar	no	no
STATBLOCK	lcurbr if while read write return id	else semi	yes	no
STATEMENTDASH	lpar equal lsqbr	else semi let if while read write return id rcurbr	no	no
APARAMS	id intlit floatlit lpar not plus minus	rpar	yes	no
ASSIGNOP	equal	id intlit floatlit lpar not plus minus	no	no
EXPR	id intlit floatlit lpar not plus minus	comma rpar semi	no	no
OPTSTRUCTDECL2	inherits	lcurbr	yes	no
REPTSTRUCTDECL4	public private	rcurbr	yes	no
STRUCTORIMPLORFUNC	struct impl func	struct impl func	no	yes
STRUCTDECL	struct	struct impl func	no	yes
IMPLDEF	impl	struct impl func	no	yes
FUNCDEF	func	struct impl func rcurbr	no	yes
TERM	id intlit floatlit lpar not plus minus	rsqbr eq neq lt gt leq geq plus minus or comma rpar semi	no	no
FACTOR	id intlit floatlit lpar not plus minus	mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar semi	no	no
RIGHTRECTERM	mult div and	rsqbr eq neq lt gt leq geq plus minus or comma rpar semi	yes	no
TYPE	integer float id	rpar lcurbr comma lsqbr semi	no	no
REPTVARDECL4	lsqbr	semi	yes	no
VARDECLORSTAT	let if while read write return id	let if while read write return id rcurbr	no	no
VARDECL	let	public private let if while read write return id rcurbr	no	no
STATEMENT	if while read write return id	else semi let if while read write return id rcurbr	no	no
VARIABLE	id	equal rpar	no	no
REPTVARIABLE2	lsqbr	mult div and rsqbr eq neq lt gt leq geq equal plus minus or comma rpar semi	yes	no
IDNEST	id	id	no	no
REPTVARIABLE0	id	id	yes	no
VISIBILITY	public private	let func	no	no

Design Overview



All the non terminals are represented as a method which is called recursively according to the token satisfying a particular grammar definition.

```
203
204     private boolean ADDOP()
205     {...}
237
238     private boolean ARITHEXPR()
239     {...}
257
258     private boolean RIGHTRECARITHEXPR()
259     {...}
278
279     private boolean RIGHTRECTERM()
280     {...}
299
300     private boolean REPTSTATBLOCK1()
301     {...}
320
321     private boolean REPTSTRUCTDECL4()
322     {...}
341
342     private boolean REPTVARDECL4()
343     {...}
362
363     private boolean ARRAYSIZE()
364     {...}
381
```

```
private boolean Match(String v)
{
    if (lookahead.equals(v))
    {
        writeOutDerivationNoLineBreak( str: lookahead+" ");
        try {
            GetNextToken();
        } catch (IOException e) {
            e.printStackTrace ();
        }
        return true;
    }
    else
    {
        writeOutSyntaxErrors( str: "Syntax error at line " +
        try {
            GetNextToken();
        } catch (IOException e) {
            e.printStackTrace ();
        }
        return false;
    }
}
```

All the terminals are caught by the Match() method and save it into the outderivation file. The methods also throws the Syntax error in the code with the appropriate suggestion.

Use of tools

1. grammartool.jar -
This tool was provided with the assignment and this tool helped me to remove almost all the ambiguities, left recursions and EBNF notations.

2. https://smlweb.cpsc.ucalgary.ca/start.html -
This tool helped me to analyze the syntactical definition and generate first and follow sets. I used this tool instead of A2CC because this tools has easy accessibility due to web interface and There is a feature to generate parse table which provides and intuition while resolving the grammar manually.