Andre Ibrahim
40132881

Section 1: LL1 Grammar

Transformed LL1 grammar in ucal format:

START -> REPTSTART0 .

APARAMS -> EXPR REPTAPARAMS1 .
APARAMS -> .

APARAMSTAIL -> comma EXPR .

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

ARITHEXPR -> TERM RIGHTRECARITHEXPR .

ARRAYSIZE -> lsqbr ARRAYSIZE2 .
ARRAYSIZE2 -> intlit rsqbr .
ARRAYSIZE2 -> rsqbr .

ASSIGNOP -> equal .

CLASSDECL -> class id OPTCLASSDECL2 lcurbr REPTCLASSDECL4 rcurbr semi .

CLASSDECLORFUNCDEF -> CLASSDECL .
CLASSDECLORFUNCDEF -> FUNCDEF .

EXPR -> ARITHEXPR EXPR2 .

EXPR2 -> RELOP ARITHEXPR .
EXPR2 -> .

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

FPARAMSTAIL -> comma id colon TYPE REPTFPARAMSTAIL4 .

FACTOR -> FUNCTIONCALLORVARIABLE .
FACTOR -> intlit .
FACTOR -> floatlit .

FACTOR -> lpar ARITHEXPR rpar .
FACTOR -> not FACTOR .
FACTOR -> SIGN FACTOR .

FUNCBODY -> lcurbr REPTFUNCBODY1 rcurbr .

FUNCDEF -> FUNCHEAD FUNCBODY .

FUNCHEAD -> function id FUNCHEAD3 .

FUNCHEAD2 -> id lpar FPARAMS rpar arrow RETURNTYPE .
FUNCHEAD2 -> constructor lpar FPARAMS rpar .

FUNCHEAD3 -> sr FUNCHEAD2 .
FUNCHEAD3 -> lpar FPARAMS rpar arrow RETURNTYPE .

ASSIGNSTAT -> VARIABLE ASSIGNOP EXPR .

FUNCTIONCALL -> id FUNCALL3 .
FUNCALL3 -> lpar APARAMS rpar .
FUNCALL3 -> FUNCALL2 .
FUNCALL2 -> dot id FUNCALL4 .
FUNCALL4 -> INDICE FUNCALL2 .
FUNCALL4 -> lpar APARAMS rpar FUNCALL5 .
FUNCALL5 -> FUNCALL2 .
FUNCALL5 -> .

VARIABLE -> id VARIABLE3 .
VARIABLE3 -> INDICE .
VARIABLE3 -> VARIABLE2 .
VARIABLE3 -> .
VARIABLE2 -> dot id VARIABLE4 .
VARIABLE4 -> lpar APARAMS rpar VARIABLE2 .
VARIABLE4 -> INDICE VARIABLE5 .
VARIABLE5 -> VARIABLE2 .
VARIABLE5 -> .

FUNCTIONCALLORVARIABLE -> id FUNCTIONCALLORVARIABLE1 .
FUNCTIONCALLORVARIABLE1 -> INDICELOOP FUNCTIONCALLORVARIABLE2 .
FUNCTIONCALLORVARIABLE1 -> lpar APARAMS rpar FUNCTIONCALLORVARIABLE2 .
FUNCTIONCALLORVARIABLE2 -> dot id FUNCTIONCALLORVARIABLE3 .
FUNCTIONCALLORVARIABLE2 -> .
FUNCTIONCALLORVARIABLE3 -> INDICELOOP FUNCTIONCALLORVARIABLE2 .
FUNCTIONCALLORVARIABLE3 -> lpar APARAMS rpar FUNCTIONCALLORVARIABLE2 .

IDNEST1 -> dot id IDNEST2 .
IDNEST2 -> lsqbr ARITHEXPR rsqbr IDNEST2 .
IDNEST2 -> lpar APARAMS rpar .
IDNEST2 -> .

INDICE -> lsqbr ARITHEXPR rsqbr .

LOCALVARDECL -> localvar id colon TYPE LOCALVARDECL2 .
LOCALVARDECL2 -> REPTLOCALVARDECL4 semi .
LOCALVARDECL2 -> lpar APARAMS rpar semi .

LOCALVARDECLORSTMT -> LOCALVARDECL .
LOCALVARDECLORSTMT -> STATEMENT .

MEMBERDECL -> MEMBERFUNCDECL .
MEMBERDECL -> MEMBERVARDECL .

MEMBERFUNCDECL -> function id colon lpar FPARAMS rpar arrow RETURNTYPE semi .
MEMBERFUNCDECL -> constructor colon lpar FPARAMS rpar semi .

MEMBERVARDECL -> attribute id colon TYPE REPTMEMBERVARDECL4 semi .

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

OPTCLASSDECL2 -> isa id REPTOPTCLASSDECL22 .
OPTCLASSDECL2 -> .

RELEXPR -> ARITHEXPR RELOP ARITHEXPR .

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

REPTSTART0 -> CLASSDECLORFUNCDEF REPTSTART0 .
REPTSTART0 -> .

REPTAPARAMS1 -> APARAMSTAIL REPTAPARAMS1 .
REPTAPARAMS1 -> .

REPTCLASSDECL4 -> VISIBILITY MEMBERDECL REPTCLASSDECL4 .
REPTCLASSDECL4 -> .

REPTFPARAMS3 -> ARRAYSIZE REPTFPARAMS3 .
REPTFPARAMS3 -> .

REPTFPARAMS4 -> FPARAMSTAIL REPTFPARAMS4 .
REPTFPARAMS4 -> .

REPTFPARAMSTAIL4 -> ARRAYSIZE REPTFPARAMSTAIL4 .
REPTFPARAMSTAIL4 -> .

REPTFUNCBODY1 -> LOCALVARDECLORSTMT REPTFUNCBODY1 .
REPTFUNCBODY1 -> .

REPTLOCALVARDECL4 -> ARRAYSIZE REPTLOCALVARDECL4 .
REPTLOCALVARDECL4 -> .

REPTMEMBERVARDECL4 -> ARRAYSIZE REPTMEMBERVARDECL4 .
REPTMEMBERVARDECL4 -> .

REPTOPTCLASSDECL22 -> comma id REPTOPTCLASSDECL22 .
REPTOPTCLASSDECL22 -> .

REPTSTATBLOCK1 -> STATEMENT REPTSTATBLOCK1 .
REPTSTATBLOCK1 -> .

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 -> FUNCTIONCALLORASIGNSTAT semi .
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 .

FUNCTIONCALLORASIGNSTAT -> id ISFUNCTIONCALLORVARIABLE .

ISFUNCTIONCALLORVARIABLE -> lpar APARAMS rpar AFTERFUNCTIONCALL .
ISFUNCTIONCALLORVARIABLE -> INDICELOOP AFTERVARIABLE .

AFTERFUNCTIONCALL -> dot id MIDDLESTATE .
AFTERVARIABLE -> dot id MIDDLESTATE .

MIDDLESTATE -> INDICELOOP AFTERVARIABLE .
MIDDLESTATE -> lpar APARAMS rpar AFTERFUNCTIONCALL .

AFTERVARIABLE -> ENDASSIGN .
AFTERFUNCTIONCALL -> .

INDICELOOP -> INDICE INDICELOOP .
INDICELOOP -> .
ENDASSIGN -> ASSIGNOP EXPR .

TERM -> FACTOR RIGHTRECTERM .

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

VISIBILITY -> public .
VISIBILITY -> private .
VISIBILITY -> .

List of left recursion and replacement:
To remove the left recursions we can use the tool given grammartool.jar and we get that we have two left recursions. Below are the left recursions found and their solution.

<term> ::= <term><multOp><factor>
replacing rules: <term>::=[<factor><rightrec-term>]

adding rules: <rightrec-term>::=[EPSILON, <multOp><factor><rightrec-term>]

<arithExpr> ::= <arithExpr><addOp><term>
replacing rules: <arithExpr>::=[<term><rightrec-arithExpr>]
adding rules: <rightrec-arithExpr>::=[EPSILON, <addOp><term><rightrec-arithExpr>]

List of ambiguities and replacement:

- ARRAYSIZE has a first set conflict.

  Old ARRAYSIZE:

  ARRAYSIZE -> lsqbr intnum rsqbr .

  ARRAYSIZE -> lsqbr rsqbr .

  New ARRAYSIZE:

  ARRAYSIZE -> lsqbr ARRAYSIZE2 .

  ARRAYSIZE2 -> intlit rsqbr .

  ARRAYSIZE2 -> rsqbr .

- EXPR has a first set conflict.

  Old EXPR:

  EXPR -> ARITHEXPR .

  EXPR -> RELEXPR .

  New Expr:

  EXPR -> ARITHEXPR EXPR2 .

  EXPR2 -> RELOP ARITHEXPR .

  EXPR2 -> .

- FACTOR has a first set conflict.

Old FACTOR:

FACTOR -> VARIABLE .

FACTOR -> FUNCTIONCALL .

FACTOR -> intlit .

FACTOR -> floatlit .

FACTOR -> lpar ARITHEXPR rpar .

FACTOR -> not FACTOR .

FACTOR -> SIGN FACTOR .

New FACTOR:

FACTOR -> FUNCTIONCALLORVARIABLE .

FACTOR -> intlit .

FACTOR -> floatlit .

FACTOR -> lpar ARITHEXPR rpar .

FACTOR -> not FACTOR .

FACTOR -> SIGN FACTOR .

FUNCTIONCALLORVARIABLE -> id FUNCTIONCALLORVARIABLE1 .

FUNCTIONCALLORVARIABLE1 -> INDICELOOP FUNCTIONCALLORVARIABLE2
.

FUNCTIONCALLORVARIABLE1 -> lpar APARAMS rpar
FUNCTIONCALLORVARIABLE2 .

FUNCTIONCALLORVARIABLE2 -> dot id FUNCTIONCALLORVARIABLE3 .

FUNCTIONCALLORVARIABLE2 -> .

FUNCTIONCALLORVARIABLE3 -> INDICELOOP FUNCTIONCALLORVARIABLE2
.

FUNCTIONCALLORVARIABLE3 -> lpar APARAMS rpar
FUNCTIONCALLORVARIABLE2 .

- FUNCHEAD has a first set conflict.

  Old FUNCHEAD:

  FUNCHEAD -> function OPTFUNCHEAD1 id lpar FPARAMS rpar arrow RETURNTYPE .

  FUNCHEAD -> function id sr constructor lpar FPARAMS rpar .

  New FUNCHEAD:

  FUNCHEAD -> function id FUNCHEAD3 .

  FUNCHEAD2 -> id lpar FPARAMS rpar arrow RETURNTYPE .

  FUNCHEAD2 -> constructor lpar FPARAMS rpar .

  FUNCHEAD3 -> sr FUNCHEAD2 .

  FUNCHEAD3 -> lpar FPARAMS rpar arrow RETURNTYPE .

- IDNEST has a first set conflict.

  Old IDNEST:

  IDNEST -> id REPTIDNEST1 dot .

  IDNEST -> id lpar APARAMS rpar dot .

  New IDNEST:

  IDNEST1 -> dot id IDNEST2 .

  IDNEST2 -> lsqbr ARITHEXPR rsqbr IDNEST2 .

  IDNEST2 -> lpar APARAMS rpar .

  IDNEST2 -> .

  ** IDNEST was changed to do .id instead of id. For the purpose of getting rid of first a follow violations the logic in all Non-terminals using IDNEST was updated

- LOCALVARDECL has a first set conflict.

  Old LOCALVARDECL:

LOCALVARDECL -> localvar id colon TYPE REPTLOCALVARDECL4 semi .

LOCALVARDECL -> localvar id colon TYPE lpar APARAMS rpar semi .

New LOCALVARDECL:

LOCALVARDECL -> localvar id colon TYPE LOCALVARDECL2 .

LOCALVARDECL2 -> REPTLOCALVARDECL4 semi .

LOCALVARDECL2 -> lpar APARAMS rpar semi .

- OPTFUNCHEAD1 is nullable with clashing first and follow sets.

  This Non Terminal was removed since the only place it was being used was in FUNCHEAD and it was redefined not to use it, please refer to the change above.

- REPTFUNCTIONCALL0 is nullable with clashing first and follow sets.

  For first and follow sets we need to see where this non-terminal is being used:

  FUNCTIONCALL -> REPTFUNCTIONCALL0 id lpar APARAMS rpar .

  Updated function call not to use this non-terminal:

  FUNCTIONCALL -> id FUNCALL3 .

  FUNCALL3 -> lpar APARAMS rpar .

  FUNCALL3 -> FUNCALL2 .

  FUNCALL2 -> dot id FUNCALL4 .

  FUNCALL4 -> INDICE FUNCALL2 .

  FUNCALL4 -> lpar APARAMS rpar FUNCALL5 .

  FUNCALL5 -> FUNCALL2 .

  FUNCALL5 -> .

- REPTVARIABLE0 is nullable with clashing first and follow sets.

  For first and follow sets we need to see where this non-terminal is being used:

  VARIABLE -> REPTVARIABLE0 id REPTVARIABLE2 .

  Updated function call not to use this non-terminal:

  VARIABLE -> id VARIABLE3 .

  VARIABLE3 -> INDICE .

  VARIABLE3 -> VARIABLE2 .

  VARIABLE3 -> .

  VARIABLE2 -> dot id VARIABLE4 .

  VARIABLE4 -> lpar APARAMS rpar VARIABLE2 .

  VARIABLE4 -> INDICE VARIABLE5 .

  VARIABLE5 -> VARIABLE2 .

  VARIABLE5 -> .

- STATEMENT has a first set conflict.

  Old STATEMENT:

  STATEMENT -> ASSIGNSTAT semi .

  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 -> FUNCTIONCALL semi .

  New STATEMENT:

STATEMENT -> FUNCTIONCALLORASIGNSTAT semi .

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 .

FUNCTIONCALLORASIGNSTAT -> id ISFUNCTIONCALLORVARIABLE .

ISFUNCTIONCALLORVARIABLE -> lpar APARAMS rpar AFTERFUNCTIONCALL .

ISFUNCTIONCALLORVARIABLE -> INDICELOOP AFTERVARIABLE .

AFTERFUNCTIONCALL -> dot id MIDDLESTATE .

AFTERVARIABLE -> dot id MIDDLESTATE .

MIDDLESTATE -> INDICELOOP AFTERVARIABLE .

MIDDLESTATE -> lpar APARAMS rpar AFTERFUNCTIONCALL .

AFTERVARIABLE -> ENDASSIGN .

AFTERFUNCTIONCALL -> .

INDICELOOP -> INDICE INDICELOOP .

INDICELOOP -> .

ENDASSIGN -> ASSIGNOP EXPR .

Section 2: first and follow sets

This is the first and follow sets for each non terminal generated by the university of calgary tool.
Note: the university of calgary tool has a nullable column which can be replaced programmatically by adding an epsilon to the first set if the field is set to “yes”

nonterminal	first set	follow set	nullable
START	class function	∅	yes
ARRAYSIZE2	intlit rsqbr	lsqbr semi rpar comma	no
CLASSDECL	class	class function	no
EXPR2	eq neq lt gt leq geq	semi comma rpar	yes
FUNCDEF	function	class function	no
FUNCBODY	lcurbr	class function	no
FUNCHEAD	function	lcurbr	no
FUNCHEAD3	sr lpar	lcurbr	no
FUNCHEAD2	id constructor	lcurbr	no
ASSIGNSTAT	id	∅	no
FUNCTIONCALL	id	∅	no
FUNCALL3	lpar dot	∅	no
FUNCALL4	lpar lsqbr	∅	no
FUNCALL5	dot	∅	yes
FUNCALL2	dot	∅	no
VARIABLE3	lsqbr dot	equal rpar	yes
VARIABLE4	lpar lsqbr	equal rpar	no
VARIABLE5	dot	equal rpar	yes
VARIABLE2	dot	equal rpar	no
FUNCTIONCALLORVARIABLE	id	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	no
FUNCTIONCALLORVARIABLE1	lpar dot lsqbr	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	yes
FUNCTIONCALLORVARIABLE3	lpar dot lsqbr	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	yes

FUNCTIONCALLORVARIABLE2	dot	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	yes
IDNEST1	dot	∅	no
IDNEST2	lsqbr lpar	∅	yes
LOCALVARDECL2	semi lpar lsqbr	localvar if while read write return id rcurbr	no
LOCALVARDECL	localvar	localvar if while read write return id rcurbr	no
MEMBERFUNCDECL	function constructor	public private function constructor attribute rcurbr	no
FPARAMS	id	rpar	yes
MEMBERVARDECL	attribute	public private function constructor attribute rcurbr	no
OPTCLASSDECL2	isa	lcurbr	yes
ARITHEXPR	intlit floatlit lpar not id plus minus	semi rsqbr eq neq lt gt leq geq comma rpar	no
RELOP	eq neq lt gt leq geq	intlit floatlit lpar not id plus minus	no
CLASSDECLORFUNCDEF	class function	class function	no
REPTSTART0	class function	∅	yes
APARAMSTAIL	comma	comma rpar	no
REPTAPARAMS1	comma	rpar	yes
MEMBERDECL	function constructor attribute	public private function constructor attribute rcurbr	no

REPTCLASSDECL4	public private function constructor attribute	rcurbr	yes
REPTFPARAMS3	lsqbr	rpar comma	yes
FPARAMSTAIL	comma	comma rpar	no
REPTFPARAMS4	comma	rpar	yes
REPTFPARAMSTAIL4	lsqbr	comma rpar	yes
LOCALVARDECLORSTMT	localvar if while read write return id	localvar if while read write return id rcurbr	no
REPTFUNCBODY1	localvar if while read write return id	rcurbr	yes
REPTLOCALVARDECL4	lsqbr	semi	yes
ARRAYSIZE	lsqbr	lsqbr semi rpar comma	no
REPTMEMBERVARDECL4	lsqbr	semi	yes
REPTOPTCLASSDECL22	comma	lcurbr	yes
RETURNTYPE	void integer float id	semi lcurbr	no
ADDOP	plus minus or	intlit floatlit lpar not id plus minus	no
RIGHTRECARITHEXPR	plus minus or	semi rsqbr eq neq lt gt leq geq comma rpar	yes
MULTOP	mult div and	intlit floatlit lpar not id plus minus	no
SIGN	plus minus	intlit floatlit lpar not id plus minus	no
REPTSTATBLOCK1	if while read write return id	rcurbr	yes

STATEMENT	if while read write return id	else semi localvar if while read write return id rcurbr	no
RELEXPR	intlit floatlit lpar not id plus minus	rpar	no
STATBLOCK	lcurbr if while read write return id	else semi	yes
VARIABLE	id	equal rpar	no
FUNCTIONCALLORASIGNSTAT	id	semi	no
ISFUNCTIONCALLORVARIABLE	lpar dot lsqbr equal	semi	no
MIDDLESTATE	lpar dot lsqbr equal	semi	no
AFTERVARIABLE	dot equal	semi	no
APARAMS	intlit floatlit lpar not id plus minus	rpar	yes
AFTERFUNCTIONCALL	dot	semi	yes
INDICE	lsqbr	semi mult div and dot lsqbr equal rsqbr eq neq lt gt leq geq plus minus or comma rpar	no
INDICELOOP	lsqbr	semi mult div and dot equal rsqbr eq neq lt gt leq geq plus minus or comma rpar	yes
ENDASSIGN	equal	semi	no
ASSIGNOP	equal	intlit floatlit lpar not id plus minus	no

EXPR	intlit floatlit lpar not id plus minus	semi comma rpar	no
TERM	intlit floatlit lpar not id plus minus	semi rsqbr eq neq lt gt leq geq plus minus or comma rpar	no
FACTOR	intlit floatlit lpar not id plus minus	semi mult div and rsqbr eq neq lt gt leq geq plus minus or comma rpar	no
RIGHTRECTERM	mult div and	semi rsqbr eq neq lt gt leq geq plus minus or comma rpar	yes
TYPE	integer float id	rpar lcurbr comma lpar lsqbr semi	no
VISIBILITY	public private	function constructor attribute	yes

Section 3: Design

The program is composed of two components, the ParsingTable and the Parser.

The responsibility of the parsing table is to import a csv file representing the parsing table into the program and store it in a way that's easy to look up. It also stores the first and following sets which will be needed to skip the errors.

The parser contains the main algorithm to check if the source file is part of the grammar defined. The .parse() method simply returns true or false based on if there was an error or not. While it runs it logs the left-most derivation and stores it so that it can be outputted in a file. It also has a skipErrors method that the parse algorithm uses in case we encounter a token that isn't valid within the grammar. The program would then go in panic mode trying to recover to the next valid token. Again, it is in this method that the first and follow sets are put to use.

Section 4: Tools

Tools used in grammar transformation:

1. grammartool.jar provided by the professor was used to remove EBNF construct and left recursions and used to translate the grammar into atocc
2. To remove ambiguities and to get an LL1 grammar I used the online atocc tool https://flaci.com/kfgedit since it was a lot more descriptive with the first and first/ first and follow conflicts

3. To get the parsing table and the first and follow sets I translated the grammar by hand from atocc to ucal format and used the university of calgary tool https://smlweb.cpsc.ucalgary.ca/start.html to get the parsing table and the first and follow sets

4. The ucal tool generates the parsing table as html and we can get that table and put it through this tool https://www.convertcsv.com/html-table-to-csv.htm to convert the tables into csv format.

   Finally with the csv format of the parsing table and first and follow sets we are ready to import them into our program so we can use them in our parsing algorithm.

Tools used in code:

1. The only tool used in the code is the Lexer that I had built in assignment 1 everything else is vanilla Typescript.