**Compiler Construction Project – Stage 1 Documentation**

**GRAMMAR**

1. <program> 🡪 <moduleDeclarations> <otherModules> <driverModule>

<otherModules>

1. <moduleDeclarations> 🡪 <moduleDeclaration><moduleDeclarations>|ε
2. <moduleDeclaration> 🡪 DECLARE MODULE ID SEMICOL
3. <otherModules> 🡪 <module><otherModules>| ε
4. <driverModule> 🡪 DRIVERDEF DRIVER PROGRAM DRIVERENDDEF<moduleDef>
5. <module> 🡪 DEF MODULE ID ENDDEF TAKES INPUT SQBO <input\_plist>

SQBC SEMICOL <ret> <moduleDef>

1. <ret> 🡪 RETURNS SQBO <output\_plist> SQBC SEMICOL | ε
2. <input\_plist> 🡪 ID COLON <dataType> <M>
3. <M> 🡪 COMMA ID COLON <dataType> <M>|ε
4. <output\_plist> 🡪 ID COLON <type> <M1>
5. <M1> 🡪 COMMA ID COLON <type> <M1>|ε
6. <dataType> --> INTEGER | REAL | BOOLEAN

| ARRAY SQBO <range> SQBC OF <type>

1. <type> 🡪 INTEGER | REAL | BOOLEAN
2. <moduleDef> 🡪 START <statements> END
3. <statements> 🡪 <statement> <statements> | ε
4. <statement> 🡪 <ioStmt>|<simpleStmt>|<declareStmt>

|<condionalStmt>|<iterativeStmt>

1. <ioStmt> 🡪 GET\_VALUE BO ID BC SEMICOL | PRINT BO <var> BC SEMICOL
2. <var> 🡪 ID <whichId> | NUM | RNUM
3. <whichId> 🡪 SQBO <index> SQBC | ε
4. <simpleStmt> 🡪 <assignmentStmt> | <moduleReuseStmt>
5. <assignmentStmt> 🡪 ID <whichStmt>
6. <whichStmt> 🡪 <lvalueIDStmt> | <lvalueARRStmt>
7. <lvalueIDStmt> 🡪 ASSIGNOP <expression> SEMICOL
8. <lvalueARRStmt> 🡪 SQBO <index> SQBC ASSIGNOP <expression> SEMICOL
9. <index> 🡪 NUM | ID
10. <moduleReuseStmt> 🡪 <optional> USE MODULE ID WITH PARAMETERS

<idList>SEMICOL

1. <optional> 🡪 SQBO <idList> SQBC ASSIGNOP | ε
2. <idList> 🡪 ID<L>
3. <L> 🡪 COMMA ID <L>|ε
4. <expression> 🡪 <arithBoolExpr>

| MINUS BO <arithmeticExpr> BC

| PLUS BO <arithmeticExpr> BC

1. <arithBoolExpr> 🡪 <select><N>

| BO <arithBoolExpr> BC

1. <N> 🡪 <logicalOp> <select> <N>|ε
2. <select> 🡪 <arithmeticExpr><N1>
3. <N1> 🡪 <relationalOp> <arithmeticExpr><N1>|ε
4. <arithmeticExpr> 🡪 <term> <K>
5. <K> 🡪 <addOp> <term><K>| ε //handling precedence and

//removing left recursion

1. <term> 🡪 <factor><R>
2. <R> 🡪 <mulOp><factor> <R>|ε //handling precedence and

//removing left recursion

1. <factor> 🡪 BO <arithmeticExpr> BC
2. <factor> 🡪 <var>
3. <addOp> 🡪 PLUS | MINUS
4. <mulOp> 🡪 MUL | DIV
5. <logicalOp> 🡪 AND | OR
6. <relationalOp> 🡪 LT | LE | GT | GE | EQ | NE
7. <declareStmt> 🡪 DECLARE <idList> COLON <dataType> SEMICOL
8. <condionalStmt> 🡪 SWITCH BO ID BC START <caseStmt><default> END
9. <caseStmt> 🡪 CASE <value> COLON <statements> BREAK SEMICOL <X1>
10. <X1> 🡪 CASE <value> COLON <statements> BREAK SEMICOL <X1> | ε
11. <value> 🡪 NUM | TRUE | FALSE
12. <default> 🡪 DEFAULT COLON <statements> BREAK SEMICOL | ε
13. <iterativeStmt> 🡪 FOR BO ID IN <range> BC START <statements> END

| WHILE BO <arithBoolExpr> BC START <statements> END

1. <range> 🡪 NUM RANGEOP NUM | ID RANGEOP ID

**FIRST SET**

1. FIRST(<range>) = {NUM,ID}
2. FIRST(<iterativeStmt>) = {FOR,WHILE}
3. FIRST(<default>) = {DEFAULT, ε}
4. FIRST(<value>) = {NUM, TRUE, FALSE}
5. FIRST(<caseStmt>) = {CASE}
6. FIRST(<X1>) = {CASE, ε}
7. FIRST(<conditionalStmt>) = {SWITCH}
8. FIRST(<declare>) = {DECLARE}
9. FIRST(<relationalOP>) = {LT, LE, GT, GE, EQ, NE}
10. FIRST(<logicalOp>) = {AND,OR}
11. FIRST(<addOp>) = {PLUS,MINUS}
12. FIRST(<mulOp>) = {MUL,DIV}
13. FIRST(<factor>) = {BO,ID,NUM,RNUM}
14. FIRST(<var>) = {ID,NUM,RNUM}
15. FIRST(<R>) = {MUL,DIV, ε}
16. FIRST(<term>) = {ID,NUM,RNUM}
17. FIRST(<K>) = {PLUS,MINUS, ε}
18. FIRST(<arithmeticExpr>) = {ID,NUM,RNUM}
19. FIRST(<expression>) = {BO,ID,NUM,RNUM,PLUS,MINUS}
20. FIRST(<arithBoolExpr>) = {BO,ID,NUM,RNUM}
21. FIRST(<N>) ={AND,OR, ε}
22. FIRST(<select>) = {BO,ID,NUM,RNUM}
23. FIRST(<N1>) = {LT,LE,GT,GE,EQ,NE, ε}
24. FIRST(<idList>) = {ID}
25. FIRST(<L>) = {COMMA, ε}
26. FIRST(<optional>) = {SQBO, ε}
27. FIRST(<moduleReuseStmt>) = {SQBO,USE}
28. FIRST(<index>) = {NUM, ID}
29. FIRST(<lvalueARRStmt>) = {SQBO}
30. FIRST(<lvalueIDStmt>) = {ASSIGNOP}
31. FIRST(<whichStmt>) = {SQBO, ASSIGNOP}
32. FIRST(<assignmentStmt>) = {ID}
33. FIRST(<simpleStmt>) = {ID, SQBO,USE}
34. FIRST(<whichId>) = {SQBO, ε}
35. FIRST(<ioStmt>) = {GET\_VALUE, PRINT}
36. FIRST(<statement>) = {GET\_VALUE,PRINT, ID,SQBO,USE, DECLARE,SWITCH,

FOR,WHILE}

1. FIRST(<statements>) = {GET\_VALUE,PRINT,ID,SQBO,USE, ε}
2. FIRST(<moduleDef>) = {START} = {DEF}
3. FIRST(<type>) = {INTEGER, REAL, BOOLEAN}
4. FIRST(<dataType>) = {INTEGER, REAL, BOOLEAN, ARRAY}
5. FIRST(<inputPlist>) = {ID}
6. FIRST(<M>) = {COMMA, ε}
7. FIRST(<outputPlist>) = {ID}
8. FIRST(<M1>) = {COMMA, ε}
9. FIRST(<ret>) = {RETURNS, ε}
10. FIRST(<module>) = {DEF}
11. FIRST(<driverModule) = {DRIVERDEF}
12. FIRST(<otherModules>) = {DEF, ε}
13. FIRST(<moduleDeclaration>) = {DECLARE}
14. FIRST(<moduleDeclarations>) = {DECLARE, ε}
15. FIRST(<program>) = {DECLARE,DEF,ε}

**FOLLOW SET**

1. FOLLOW(<program>) = {$}
2. FOLLOW(<moduleDeclarations>) = {DEF, DRIVERDEF}
3. FOLLOW(<moduleDeclaration>) = {DEF,DRIVERDEF}
4. FOLLOW(<otherModules>) = {DEF,$}
5. FOLLOW(<driverModule>) = {DEF,$}
6. FOLLOW(<module>) = {DEF,$}
7. FOLLOW(<ret>) = {START}
8. FOLLOW(<M>) = {SQBC}
9. FOLLOW(<input\_plist>) = {SQBC}
10. FOLLOW(<M1>) = {SQBC}
11. FOLLOW(<output\_plist>) = {SQBC}
12. FOLLOW(<dataType>) = {COMMA,SEMICOL,SQBC}
13. FOLLOW(<type>) = {COMMA,SEMICOL,SQBC}
14. FOLLOW(<statements>)= {END}
15. FOLLOW(<moduleDef>) = {DEF,DECLARE,$}
16. FOLLOW(<whichId>) = {BC}
17. FOLLOW(<statements>) = {END, BREAK}
18. FOLLOW(<statement>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END, BREAK}
19. FOLLOW(<ioStmt>)= {GET\_VALUE, PRINT, ID, SQBO, USE, END, BREAK}
20. FOLLOW(<whichStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END, BREAK}
21. FOLLOW(<simpleStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<assignmentStmt>)= {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<lvalueIDStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<lvalueARRStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<moduleReuseStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<declareStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<contitionalStmt>) = {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<optional>) = {USE}
2. FOLLOW(<L>) = {SEMICOL,SQBC,COLON,COMMA}
3. FOLLOW(<idList>) = {SEMICOL,SQBC,COLON}
4. FOLLOW(<N1>) = {AND, OR, SEMICOL, BC}
5. FOLLOW(<select>) = {AND,OR,SEMICOL,BC}
6. FOLLOW(<N>) = {SEMICOL}
7. FOLLOW(<arithBoolExpr>) = {SEMICOL}
8. FOLLOW(<arithmeticExpr>) = {LT,LE,GT,GE,EQ,NE,SEMICOL,BC}
9. FOLLOW(<K>) = {LT,LE,GT,GE,EQ,NE,SEMICOL,BC}
10. FOLLOW(<R>) = {PLUS,MINUS,LT,LE,GT,GE,EQ,NE,SEMICOL,BC}
11. FOLLOW(<term>= {PLUS,MINUS,LT,LE,GT,GE,EQ,NE,SEMICOL,BC}
12. FOLLOW(<var>)= {BC,MUL,DIV,PLUS,MINUS,LT,LE,GT,GE,EQ,NE,SEMICOL}
13. FOLLOW(<factor>)= {BC,MUL,DIV,PLUS,MINUS,LT,LE,GT,GE,EQ,NE,SEMICOL}
14. FOLLOW(<mulOp>) = {BO,ID,NUM,RNUM}
15. FOLLOW(<addOp>) = {BO,ID,NUM,RNUM}
16. FOLLOW(<logicalOp>)= {BO,ID,NUM,RNUM}
17. FOLLOW(<relationalOp) = {BO,ID,NUM,RNUM}
18. FOLLOW(<caseStmt>)= {DEFAULT}
19. FOLLOW(<value>)= {COLON}
20. FOLLOW(<default>)= {END}
21. FOLLOW(<X1>)= {DEFAULT}
22. FOLLOW(<iterativeStmt>)= {GET\_VALUE, PRINT, ID, SQBO, USE, END,

BREAK}

1. FOLLOW(<range>) = {SQBC,BC}

Ayush Vachaspati – 2016B3A70398P

Indraneel Ghosh – 2016B1A70938P

G Adityan – 2016B1A70929P