PLAN (AKA PROGRAMMING - LANGUAGE)

SER 502 PROJECT

Nihal Singh - 1223932476 - nnolas27

Venkata Kanaka Rama Meher Virinchi Gudimetla-1223592279- vgudimet

Sai Chandra Kaushik Reddy Parvatala - 1224356627- sparvat6

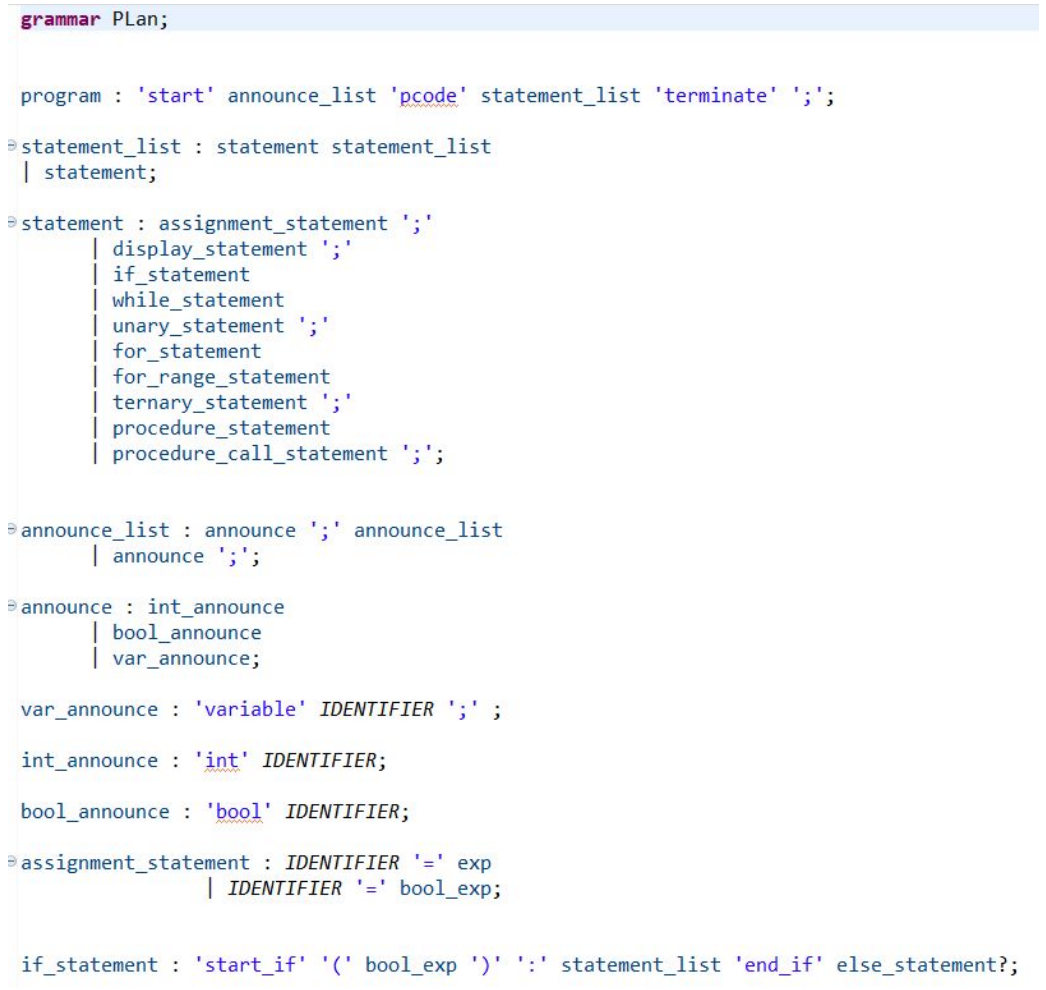
Prakruthi Ravandur Madesh - 1211219734 - pravandu

Anila Devarashetty - 1222977366 - adevaras

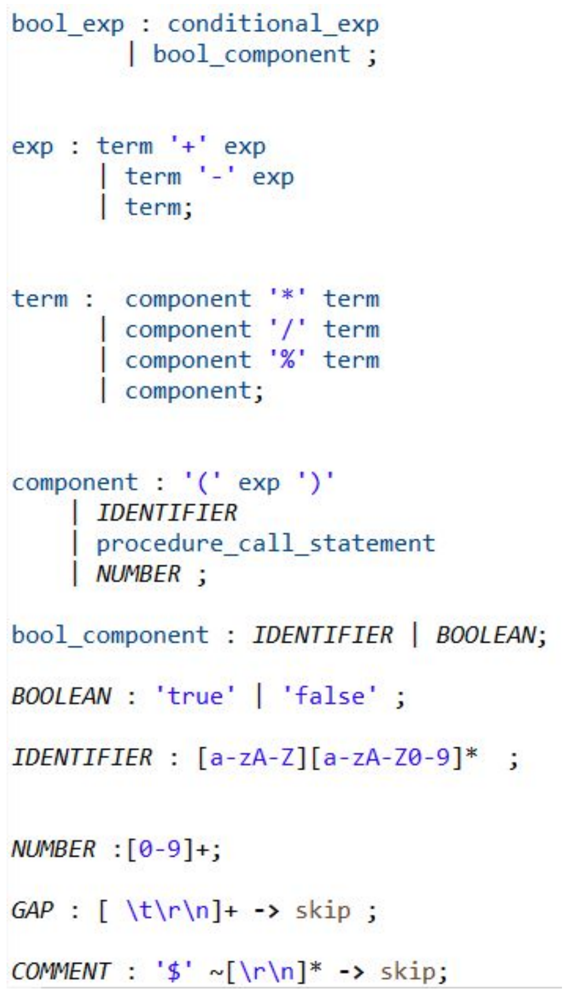
SUMMARY OF PLAN

- PLan also called as Programming– Language is a simple programming language that we have made using ANTLR4 and Java 8, it works with a .plan extension and has the ability to perform arithmetic operations and expressions including traditional iterations, conditions and recursive loops.

GRAMMAR - I

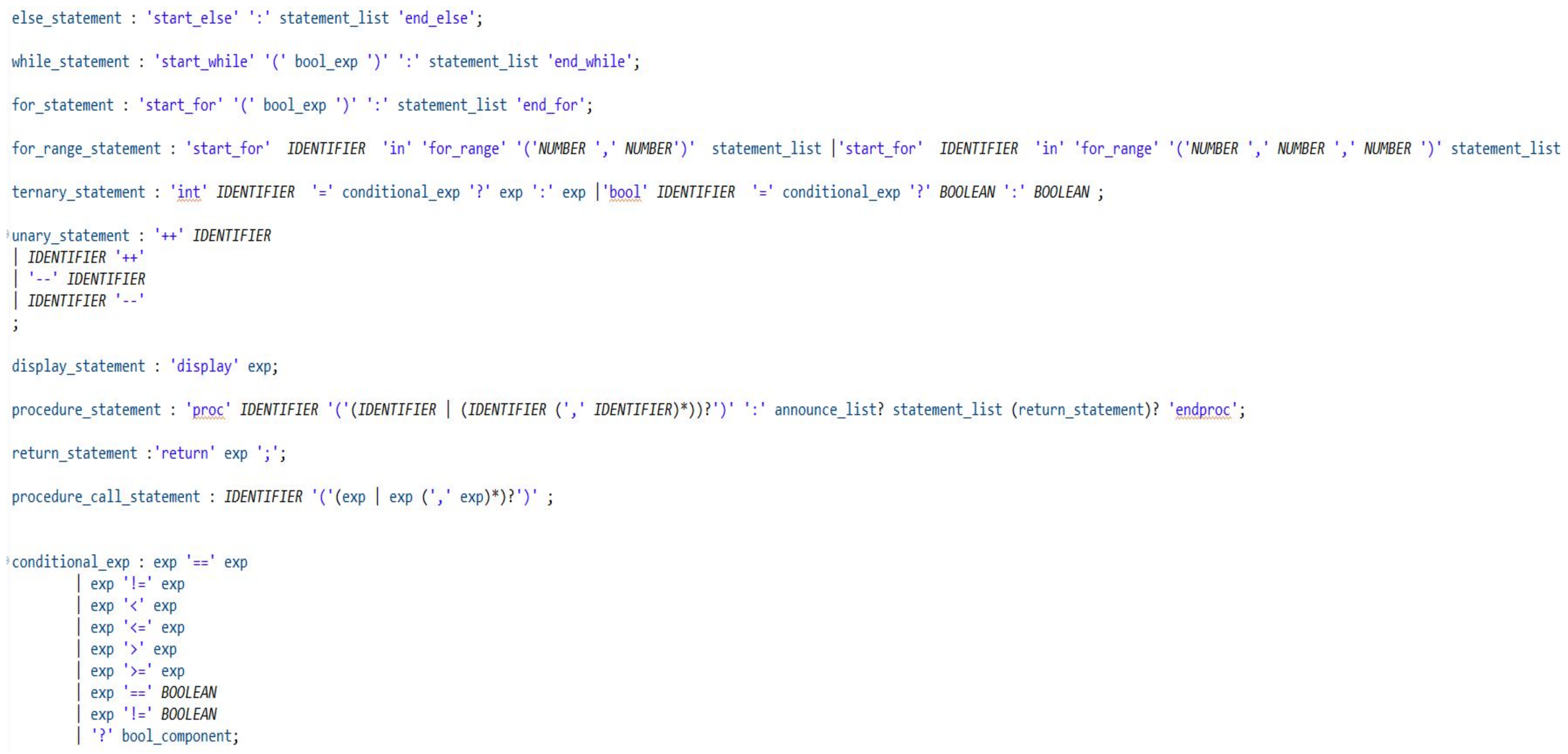


```
grammar PLan;


program : 'start' announce_list 'pcode' statement_list 'terminate' ';';

statement_list : statement statement_list
| statement;

statement : assignment_statement ';'
        | display_statement ';'
        | if_statement
        | while_statement
        | unary_statement ';'
        | for_statement
        | for_range_statement
        | ternary_statement ';'
        | procedure_statement
        | procedure_call_statement ';';


announce_list : announce ';' announce_list
        | announce ';';

announce : int_announce
        | bool_announce
        | var_announce;

var_announce : 'variable' IDENTIFIER ';' ;

int_announce : 'int' IDENTIFIER;

bool_announce : 'bool' IDENTIFIER;

assignment_statement : IDENTIFIER '=' exp
                | IDENTIFIER '=' bool_exp;


if_statement : 'start_if' '(' bool_exp ')' ':' statement_list 'end_if' else_statement?;
```



```
bool_exp : conditional_exp
        | bool_component ;


exp : term '+' exp
      | term '-' exp
      | term;


term :  component '*' term
      | component '/' term
      | component '%' term
      | component;


component : '(' exp ')'
     | IDENTIFIER
     | procedure_call_statement
     | NUMBER ;

bool_component : IDENTIFIER | BOOLEAN;

BOOLEAN : 'true' | 'false' ;

IDENTIFIER : [a-zA-Z][a-zA-Z0-9]*  ;


NUMBER :[0-9]+;

GAP : [ \t\r\n]+ -> skip ;

COMMENT : '$' ~[\r\n]* -> skip;
```

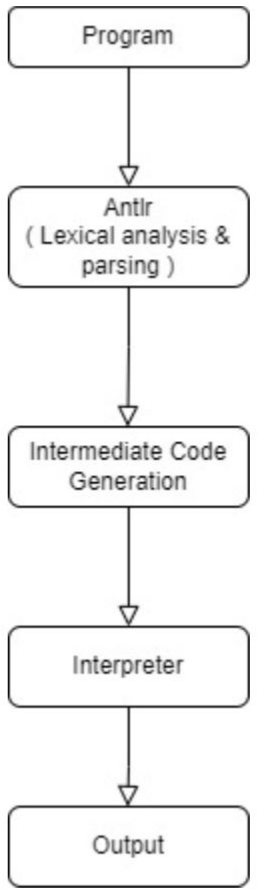
GRAMMAR - II



```
else_statement : 'start_else' ':' statement_list 'end_else';

while_statement : 'start_while' '(' bool_exp ')' ':' statement_list 'end_while';

for_statement : 'start_for' '(' bool_exp ')' ':' statement_list 'end_for';

for_range_statement : 'start_for'  IDENTIFIER  'in' 'for_range' '('NUMBER ',' NUMBER')'  statement_list |'start_for'  IDENTIFIER  'in' 'for_range' '('NUMBER ',' NUMBER ',' NUMBER ')' statement_list

ternary_statement : 'int' IDENTIFIER  '=' conditional_exp '?' exp ':' exp |'bool' IDENTIFIER  '=' conditional_exp '?' BOOLEAN ':' BOOLEAN ;

unary_statement : '++' IDENTIFIER
| IDENTIFIER '++'
| '--' IDENTIFIER
| IDENTIFIER '--'
;

display_statement : 'display' exp;

procedure_statement : 'proc' IDENTIFIER '('(IDENTIFIER | (IDENTIFIER (',' IDENTIFIER)*))?')' ':' announce_list? statement_list (return_statement)? 'endproc';

return_statement :'return' exp ';';

procedure_call_statement : IDENTIFIER '('(exp | exp (',' exp)*)?')' ;


conditional_exp : exp '==' exp
        | exp '!=' exp
        | exp '<' exp
        | exp '<=' exp
        | exp '>' exp
        | exp '>=' exp
        | exp '==' BOOLEAN
        | exp '!=' BOOLEAN
        | '?' bool_component;
```

STRUCTURE OF PLAN

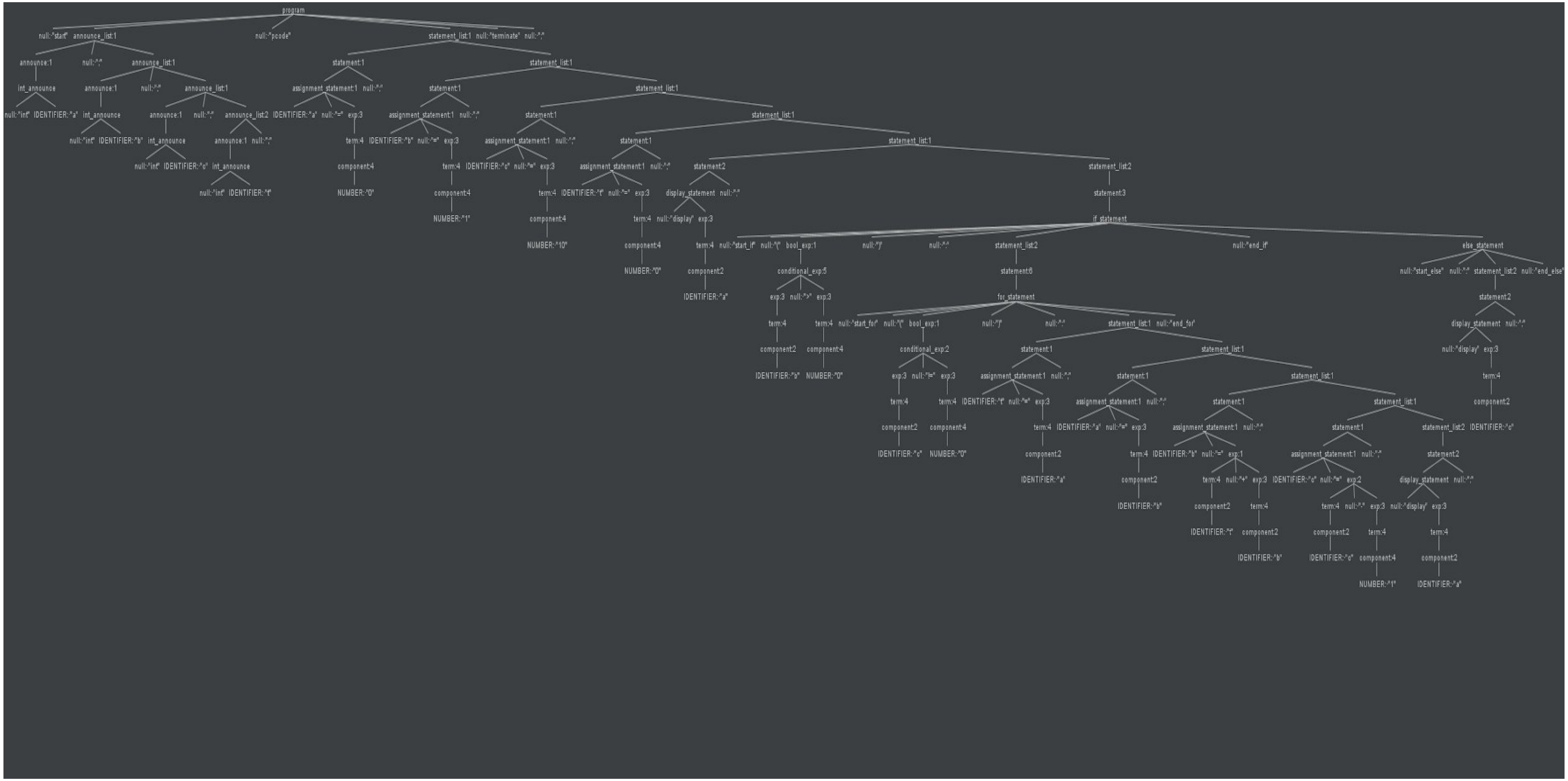
- PLan follows a simple structure with a modular flow, the initial program that we write is scanned by the lexical analyzer and generates tokens when requested by the parser and an intermediate code is generated. The interpreter then uses this to verify whether the code is semantically accurate or not and further generates the output after program evaluation.


Program
Antlr (Lexical analysis & parsing)
Intermediate Code Generation
Interpreter
Output

TOOLS AND ENVIRONMENTS

- ANTLR4 – Another Tool for Language Recognition (Version 4.0)
- JAVA – Intermediate Code Generation, Run time Environment.
- Eclipse – IDE, Run time environment.

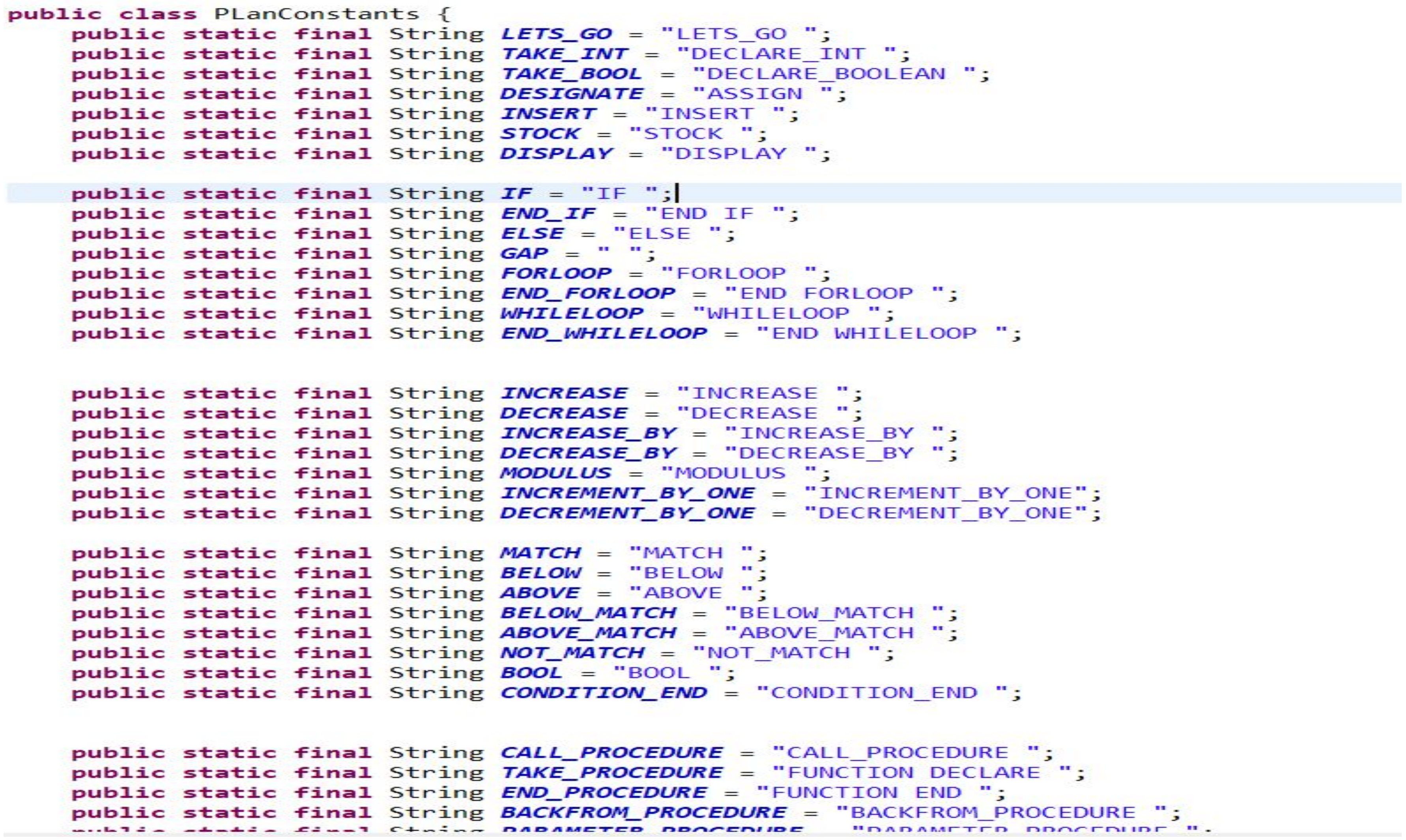
PARSE TREE


program
null:"start" announce_list:1 null:"pcode" statement_list:1 null:"terminate" null:";"
announce:1 null:";" announce_list:1 statement:1 statement_list:1
int_announce announce:1 null:";" announce_list:1 assignment_statement:1 null:";" statement:1 statement_list:1
null:"int" IDENTIFIER:"a" int_announce announce:1 null:";" announce_list:2 IDENTIFIER:"a" null:"=" exp:3 assignment_statement:1 null:";" statement:1 statement_list:1
null:"int" IDENTIFIER:"b" int_announce announce:1 null:";" term:4 IDENTIFIER:"b" null:"=" exp:3 assignment_statement:1 null:";" statement:1 statement_list:1
null:"int" IDENTIFIER:"c" int_announce component:4 term:4 IDENTIFIER:"c" null:"=" exp:3 assignment_statement:1 null:";" statement:2 statement_list:2
null:"int" IDENTIFIER:"f" NUMBER:"0" component:4 term:4 IDENTIFIER:"f" null:"=" exp:3 display_statement null:";" statement:3
NUMBER:"1" component:4 term:4 null:"display" exp:3 if_statement
NUMBER:"10" component:4 term:4 null:"start_if" null:"(" bool_exp:1 null:")" null:":" statement_list:2 null:"end_if" else_statement
NUMBER:"0" component:2 conditional_exp:5 statement:6 null:"start_else" null:":" statement_list:2 null:"end_else"
IDENTIFIER:"a" exp:3 null:">" exp:3 for_statement statement:2
term:4 term:4 null:"start_for" null:"(" bool_exp:1 null:")" null:":" statement_list:1 null:"end_for" display_statement null:";"
component:2 component:4 conditional_exp:2 statement:1 statement_list:1 null:"display" exp:3
IDENTIFIER:"b" NUMBER:"0" exp:3 null:"!=" exp:3 assignment_statement:1 null:";" statement:1 statement_list:1 term:4
term:4 term:4 IDENTIFIER:"f" null:"=" exp:3 assignment_statement:1 null:";" statement:1 statement_list:1 component:2
component:2 component:4 term:4 IDENTIFIER:"a" null:"=" exp:3 assignment_statement:1 null:";" statement:1 statement_list:2 IDENTIFIER:"c"
IDENTIFIER:"c" NUMBER:"0" component:2 term:4 IDENTIFIER:"b" null:"=" exp:1 assignment_statement:1 null:";" statement:2
IDENTIFIER:"a" component:2 term:4 null:"+" exp:3 IDENTIFIER:"c" null:"=" exp:2 display_statement null:";"
IDENTIFIER:"b" component:2 term:4 term:4 null:"-" exp:3 null:"display" exp:3
IDENTIFIER:"f" component:2 component:2 term:4 term:4
IDENTIFIER:"b" IDENTIFIER:"c" component:4 component:2
NUMBER:"1" IDENTIFIER:"a"

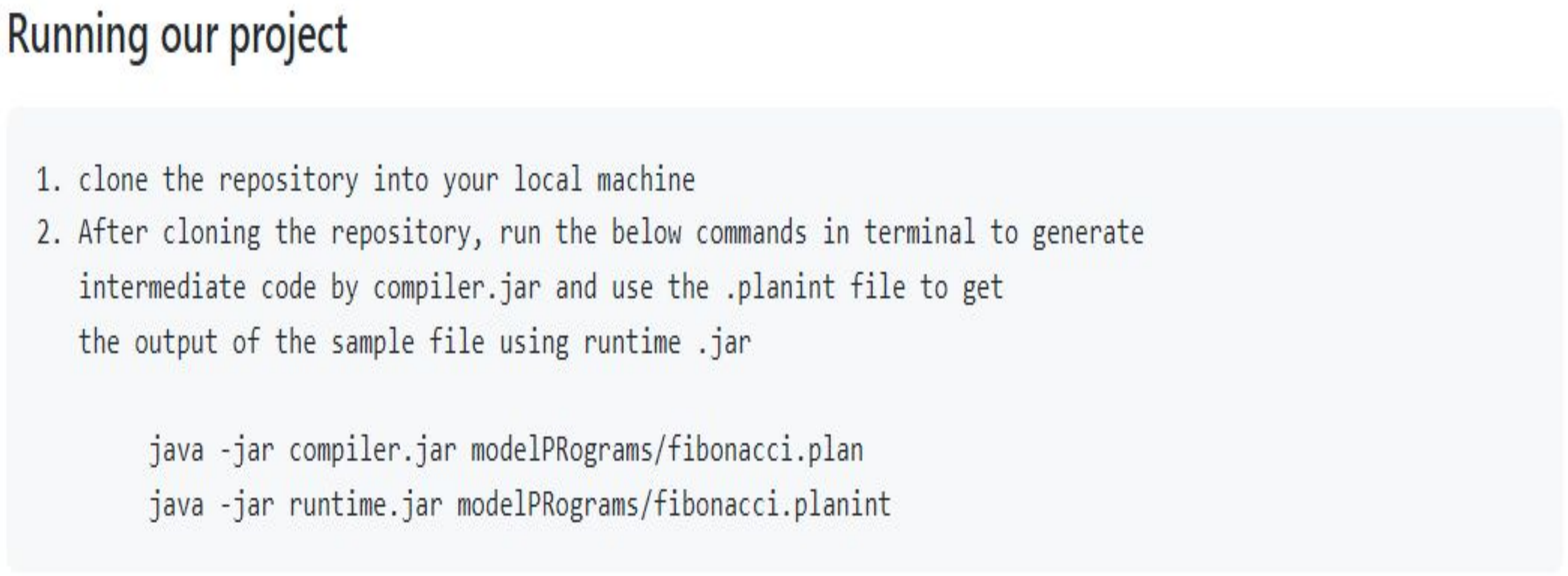
INTERMEDIATE CODE GENERATION – WORK FLOW

- ANTLR4 generates an interface called PLanListener which is further implemented by the PLanBaseListener.
- The PLanBaseListener scans the the parse tree which is generated after the lexer creates tokens for the input.
- The PLanBaseListener handles the corresponding parse rule after scanning the parse tree.
- The PLanBaseListener class extends for Intermediate code generation.

CONSTANT MAPPING FOR RUNTIME AND INTERMEDIATE CODE GENERATION



```
public class PLanConstants {
    public static final String LETS_GO = "LETS_GO ";
    public static final String TAKE_INT = "DECLARE_INT ";
    public static final String TAKE_BOOL = "DECLARE_BOOLEAN ";
    public static final String DESIGNATE = "ASSIGN ";
    public static final String INSERT = "INSERT ";
    public static final String STOCK = "STOCK ";
    public static final String DISPLAY = "DISPLAY ";

    public static final String IF = "IF ";
    public static final String END_IF = "END IF ";
    public static final String ELSE = "ELSE ";
    public static final String GAP = " ";
    public static final String FORLOOP = "FORLOOP ";
    public static final String END_FORLOOP = "END FORLOOP ";
    public static final String WHILELOOP = "WHILELOOP ";
    public static final String END_WHILELOOP = "END WHILELOOP ";


    public static final String INCREASE = "INCREASE ";
    public static final String DECREASE = "DECREASE ";
    public static final String INCREASE_BY = "INCREASE_BY ";
    public static final String DECREASE_BY = "DECREASE_BY ";
    public static final String MODULUS = "MODULUS ";
    public static final String INCREMENT_BY_ONE = "INCREMENT_BY_ONE";
    public static final String DECREMENT_BY_ONE = "DECREMENT_BY_ONE";

    public static final String MATCH = "MATCH ";
    public static final String BELOW = "BELOW ";
    public static final String ABOVE = "ABOVE ";
    public static final String BELOW_MATCH = "BELOW_MATCH ";
    public static final String ABOVE_MATCH = "ABOVE_MATCH ";
    public static final String NOT_MATCH = "NOT_MATCH ";
    public static final String BOOL = "BOOL ";
    public static final String CONDITION_END = "CONDITION_END ";


    public static final String CALL_PROCEDURE = "CALL_PROCEDURE ";
    public static final String TAKE_PROCEDURE = "FUNCTION DECLARE ";
    public static final String END_PROCEDURE = "FUNCTION END ";
    public static final String BACKFROM_PROCEDURE = "BACKFROM_PROCEDURE ";
```



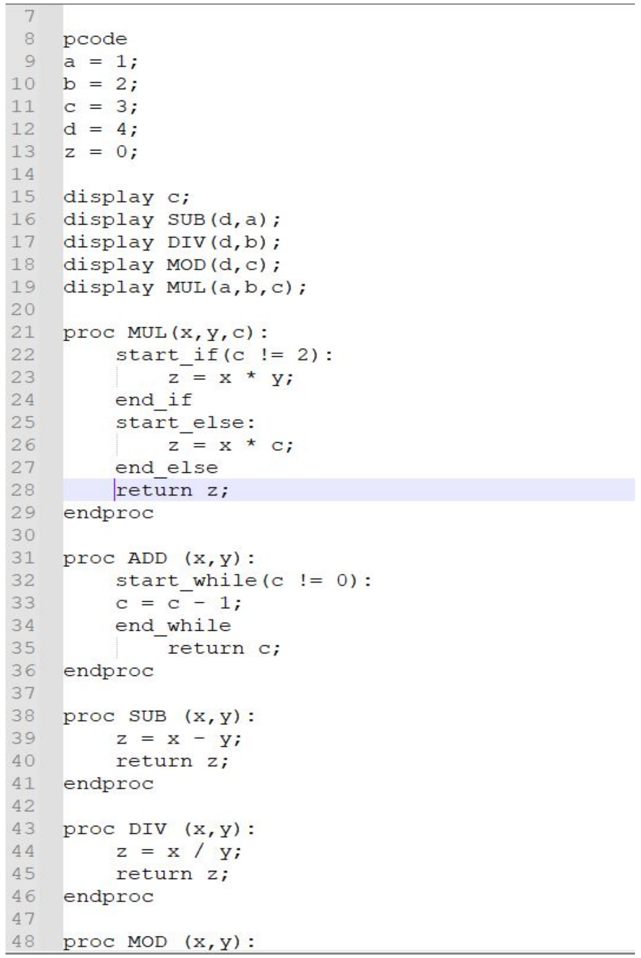
Running our project

1. clone the repository into your local machine
2. After cloning the repository, run the below commands in terminal to generate
   intermediate code by compiler.jar and use the .planint file to get
   the output of the sample file using runtime .jar

```
java -jar compiler.jar modelPRograms/fibonacci.plan
java -jar runtime.jar modelPRograms/fibonacci.planint
```

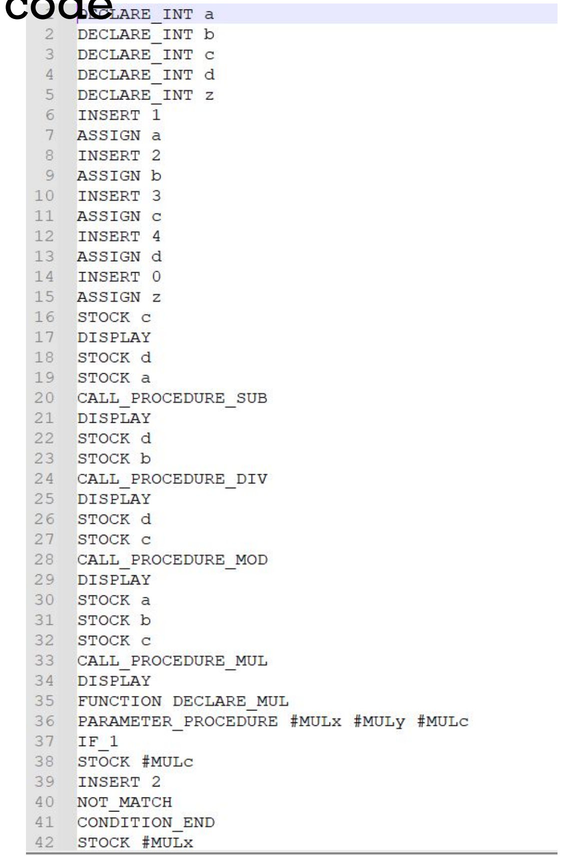
SAMPLE PROGRAM 1 – FUNCTIONAL PROCEDURE

Sample Code



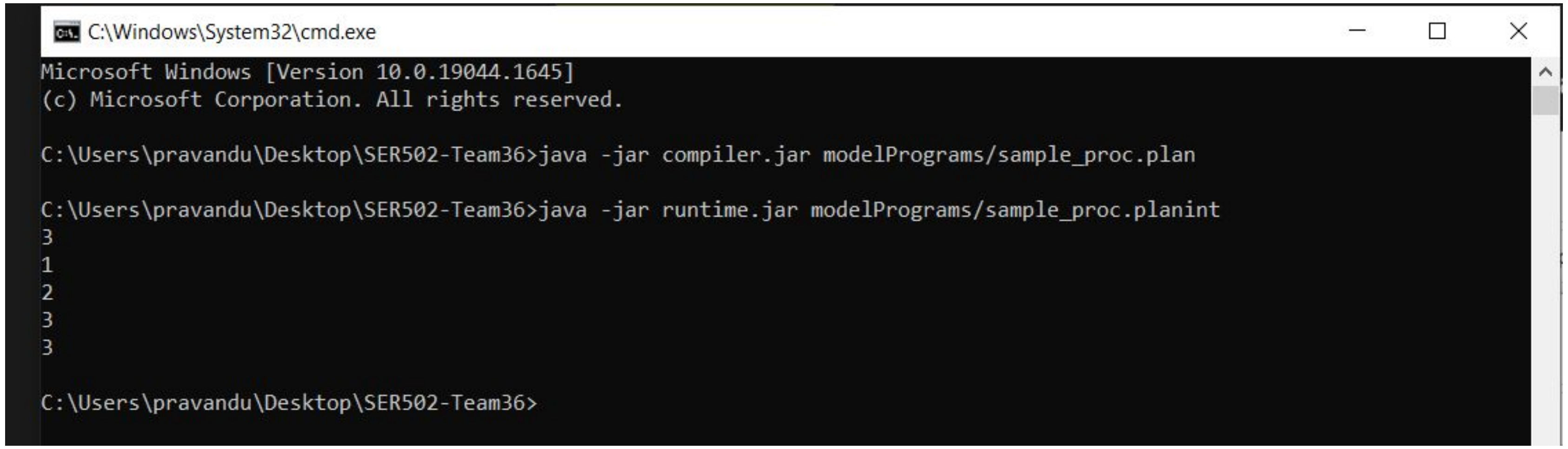
```

pcode
a = 1;
b = 2;
c = 3;
d = 4;
z = 0;

display c;
display SUB(d,a);
display DIV(d,b);
display MOD(d,c);
display MUL(a,b,c);

proc MUL(x,y,c):
    start_if(c != 2):
        z = x * y;
    end_if
    start_else:
        z = x * c;
    end_else
    return z;
endproc

proc ADD (x,y):
    start_while(c != 0):
    c = c - 1;
    end_while
        return c;
endproc

proc SUB (x,y):
    z = x - y;
    return z;
endproc

proc DIV (x,y):
    z = x / y;
    return z;
endproc

proc MOD (x,y):
```

Intermediate code



```
DECLARE_INT a
DECLARE_INT b
DECLARE_INT c
DECLARE_INT d
DECLARE_INT z
INSERT 1
ASSIGN a
INSERT 2
ASSIGN b
INSERT 3
ASSIGN c
INSERT 4
ASSIGN d
INSERT 0
ASSIGN z
STOCK c
DISPLAY
STOCK d
STOCK a
CALL_PROCEDURE_SUB
DISPLAY
STOCK d
STOCK b
CALL_PROCEDURE_DIV
DISPLAY
STOCK d
STOCK c
CALL_PROCEDURE_MOD
DISPLAY
STOCK a
STOCK b
STOCK c
CALL_PROCEDURE_MUL
DISPLAY
FUNCTION DECLARE_MUL
PARAMETER_PROCEDURE #MULx #MULy #MULc
IF_1
STOCK #MULc
INSERT 2
NOT_MATCH
CONDITION_END
STOCK #MULx
```

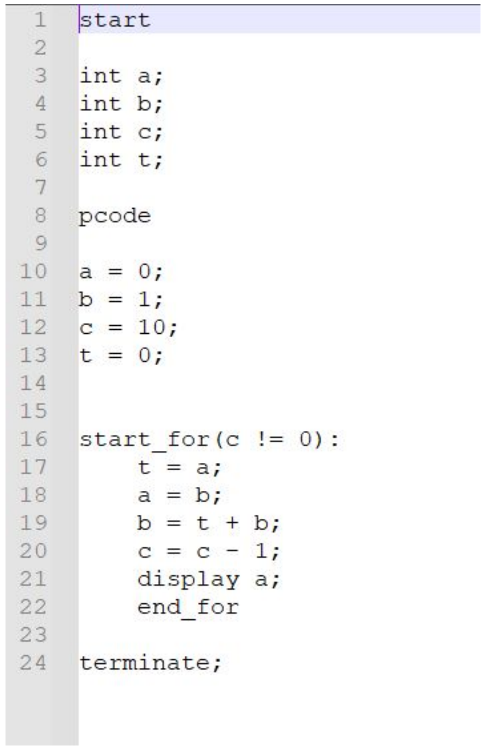
Outpu
t



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\Users\pravandu\Desktop\SER502-Team36>java -jar compiler.jar modelPrograms/sample_proc.plan

C:\Users\pravandu\Desktop\SER502-Team36>java -jar runtime.jar modelPrograms/sample_proc.planint
3
1
2
3
3

C:\Users\pravandu\Desktop\SER502-Team36>
```

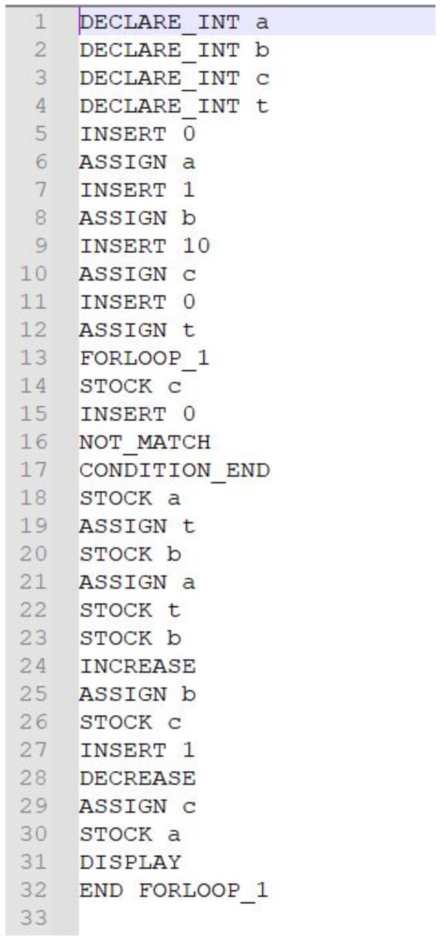
SAMPLE PROGRAM 2 – FIBONACCI SERIES

Sample Code



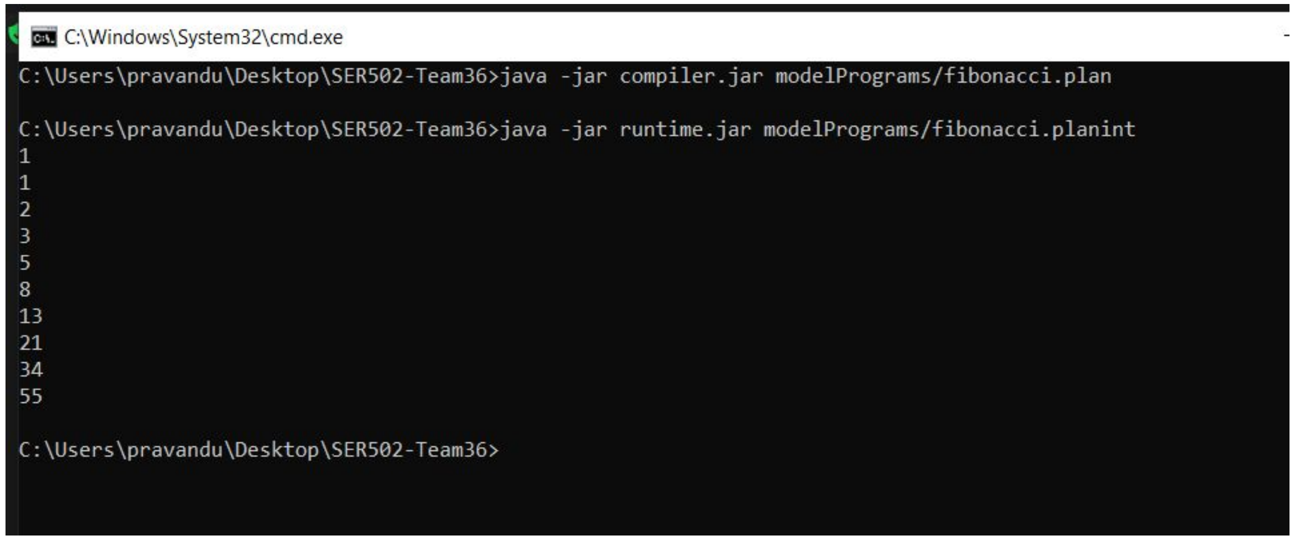
```
start

int a;
int b;
int c;
int t;

pcode

a = 0;
b = 1;
c = 10;
t = 0;


start_for(c != 0):
    t = a;
    a = b;
    b = t + b;
    c = c - 1;
    display a;
    end_for

terminate;
```

Intermediate code



```
DECLARE_INT a
DECLARE_INT b
DECLARE_INT c
DECLARE_INT t
INSERT 0
ASSIGN a
INSERT 1
ASSIGN b
INSERT 10
ASSIGN c
INSERT 0
ASSIGN t
FORLOOP_1
STOCK c
INSERT 0
NOT_MATCH
CONDITION_END
STOCK a
ASSIGN t
STOCK b
ASSIGN a
STOCK t
STOCK b
INCREASE
ASSIGN b
STOCK c
INSERT 1
DECREASE
ASSIGN c
STOCK a
DISPLAY
END FORLOOP_1

```

Outpu
t

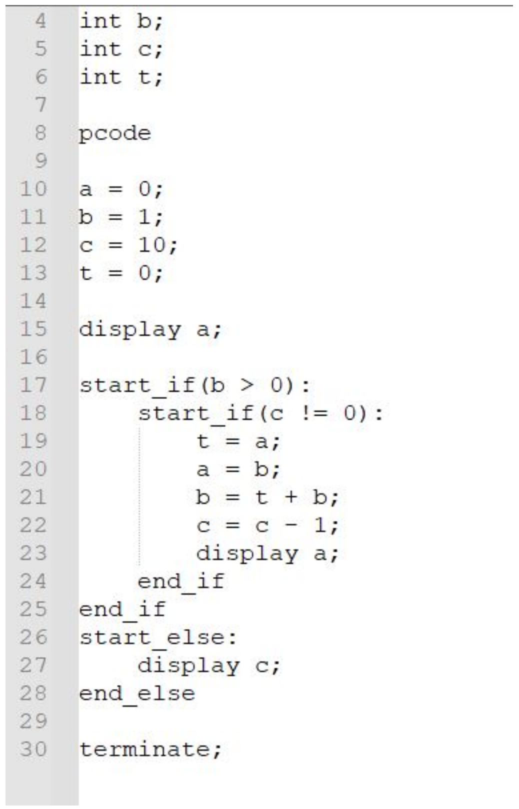


C:\Windows\System32\cmd.exe

```
C:\Users\pravandu\Desktop\SER502-Team36>java -jar compiler.jar modelPrograms/fibonacci.plan

C:\Users\pravandu\Desktop\SER502-Team36>java -jar runtime.jar modelPrograms/fibonacci.planint
1
1
2
3
5
8
13
21
34
55

C:\Users\pravandu\Desktop\SER502-Team36>
```

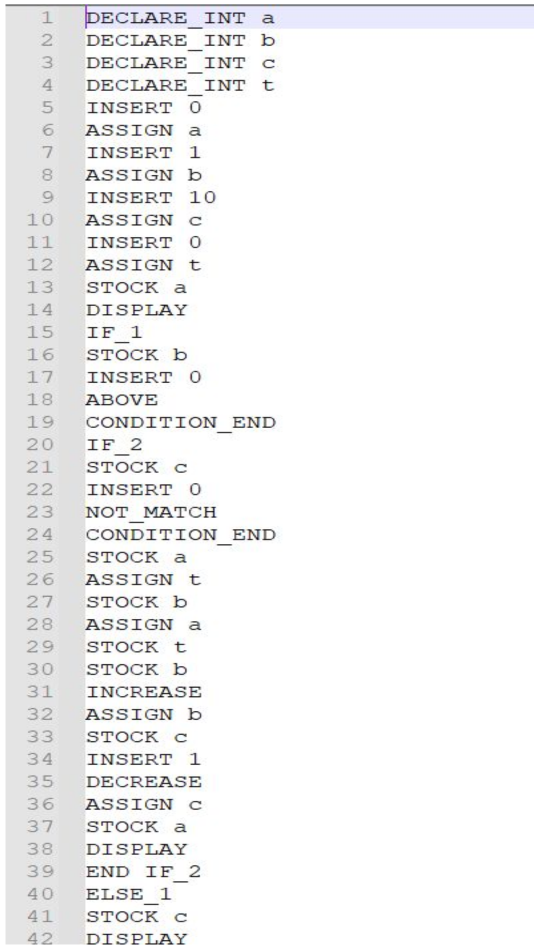
SAMPLE PROGRAM 3 – NESTED-IF

Sample Code



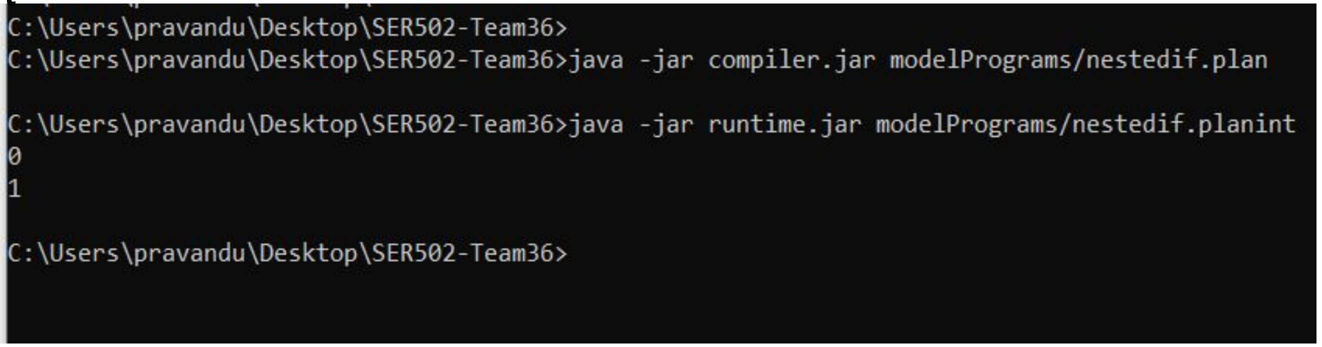
```
int b;
int c;
int t;

pcode

a = 0;
b = 1;
c = 10;
t = 0;

display a;

start_if(b > 0):
    start_if(c != 0):
        t = a;
        a = b;
        b = t + b;
        c = c - 1;
        display a;
    end_if
end_if
start_else:
    display c;
end_else

terminate;
```

Intermediate code



```
DECLARE_INT a
DECLARE_INT b
DECLARE_INT c
DECLARE_INT t
INSERT 0
ASSIGN a
INSERT 1
ASSIGN b
INSERT 10
ASSIGN c
INSERT 0
ASSIGN t
STOCK a
DISPLAY
IF_1
STOCK b
INSERT 0
ABOVE
CONDITION_END
IF_2
STOCK c
INSERT 0
NOT_MATCH
CONDITION_END
STOCK a
ASSIGN t
STOCK b
ASSIGN a
STOCK t
STOCK b
INCREASE
ASSIGN b
STOCK c
INSERT 1
DECREASE
ASSIGN c
STOCK a
DISPLAY
END_IF_2
ELSE_1
STOCK c
DISPLAY
```

Outpu
t



```
C:\Users\pravandu\Desktop\SER502-Team36>
C:\Users\pravandu\Desktop\SER502-Team36>java -jar compiler.jar modelPrograms/nestedif.plan

C:\Users\pravandu\Desktop\SER502-Team36>java -jar runtime.jar modelPrograms/nestedif.planint
0
1

C:\Users\pravandu\Desktop\SER502-Team36>
```

FUTURE GOALS

- Enable string operations and data manipulation in string.
- Support for data structures like array, stack and queue.
- Support of external libraries.

YOUTUBE VIDEO LINK:

https://youtu.be/Da0idxR8HJQ

THANK YOU