CMPT 432 - Spring 2021 - Dr. Labouseur

# Code Generation Testing Results

Emily Doran
Emily.Doran1@Marist.edu

May 24, 2021

## Single Scope Test Cases (Verbose Mode)

**Input text:**

```
{
  int a
  a = 1
  print(a)
}$

{
  int i
  int j
  i = 9
  j = 5
  print(i)
  print(j)
}$

{
  string n
  n = "emily"
  print(n)
}$

{
  boolean b
  b = false
  print(b)
}$
```

**Output:**

```
INFO  Lexer - Lexing program 1...
```

```
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (2:3)
DEBUG Lexer - T_ID [ a ] found at (2:7)
DEBUG Lexer - T_ID [ a ] found at (3:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (3:5)
DEBUG Lexer - T_DIGIT [ 1 ] found at (3:7)
DEBUG Lexer - T_PRINT [ print ] found at (4:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (4:8)
DEBUG Lexer - T_ID [ a ] found at (4:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (4:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (5:1)
DEBUG Lexer - T_EOP [ $ ] found at (5:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 1 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[a]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[a]
------[=]
------<Expression>
-------<IntegerExpression>
--------<Digit>
---------[1]
----<StatementList>
-----<Statement>
------<PrintStatement>
-------[print]
-------[(]
-------<Expression>
--------<Id>
---------[a]
-------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 1 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 1.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (2:3)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (3:7)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (4:9)

Program 1 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 1 ...
<BLOCK>
-<VariableDeclaration>
--[int]
--[a]
-<Assign>
--[a]
--[1]
-<Print>
--[a]

Program 1 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
a     int      0      2


CODE GENERATION: Beginning Code Generation on Program 1 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: a
CODE GENERATION: Assigning Variable a to value: 1
CODE GENERATION: Printing variable: a
CODE GENERATION: Adding Break Statement
CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
11
Program 1 Code Generation Passed With 0 error(s)

Program 1 Static Variable Table
---------------------------------
Name  Temp   Address  Scope
---------------------------------
a     T0XX   11        0

Program 1 Jump Table
----------------------
```

```
Temp  Distance
----------------------

Program 1 Machine Code:
A9 00 8D 11 00 A9 01 8D 11 00 AC 11 00 A2 01 FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 74 72 75 65 00 66 61 6C 73 65 00



INFO  Lexer - Lexing program 2...
DEBUG Lexer - T_L_BRACE [ { ] found at (7:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (8:3)
DEBUG Lexer - T_ID [ i ] found at (8:7)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (9:3)
DEBUG Lexer - T_ID [ j ] found at (9:7)
DEBUG Lexer - T_ID [ i ] found at (10:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (10:5)
DEBUG Lexer - T_DIGIT [ 9 ] found at (10:7)
DEBUG Lexer - T_ID [ j ] found at (11:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (11:5)
DEBUG Lexer - T_DIGIT [ 5 ] found at (11:7)
DEBUG Lexer - T_PRINT [ print ] found at (12:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (12:8)
DEBUG Lexer - T_ID [ i ] found at (12:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (12:10)
DEBUG Lexer - T_PRINT [ print ] found at (13:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (13:8)
DEBUG Lexer - T_ID [ j ] found at (13:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (13:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (14:1)
DEBUG Lexer - T_EOP [ $ ] found at (14:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 2 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
```

```
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 2 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[i]
---<StatementList>
----<Statement>
-----<VarDecl>
------<Type>
-------[int]
------<Id>
-------[j]
----<StatementList>
-----<Statement>
------<AssignStatement>
-------<Id>
--------[i]
-------[=]
-------<Expression>
--------<IntegerExpression>
---------<Digit>
----------[9]
-----<StatementList>
------<Statement>
-------<AssignStatement>
--------<Id>
---------[j]
--------[=]
--------<Expression>
---------<IntegerExpression>
----------<Digit>
-----------[5]
------<StatementList>
-------<Statement>
--------<PrintStatement>
---------[print]
---------[(]
---------<Expression>
----------<Id>
-----------[i]
---------[)]
-------<StatementList>
--------<Statement>
---------<PrintStatement>
----------[print]
----------[(]
----------<Expression>
-----------<Id>
------------[j]
----------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 2 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 7.
SEMANTIC ANALYSIS: Variable [ i ] has been declared at (8:3)
SEMANTIC ANALYSIS: Variable [ j ] has been declared at (9:3)
SEMANTIC ANALYSIS: Variable [ i ] has been initialized at (10:7)
SEMANTIC ANALYSIS: Variable [ j ] has been initialized at (11:7)
SEMANTIC ANALYSIS: Variable [ i ] has been used at (12:9)
SEMANTIC ANALYSIS: Variable [ j ] has been used at (13:9)

Program 2 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 2 ...
<BLOCK>
-<VariableDeclaration>
--[int]
--[i]
-<VariableDeclaration>
--[int]
--[j]
-<Assign>
--[i]
--[9]
-<Assign>
--[j]
--[5]
-<Print>
--[i]
-<Print>
--[j]
```

```
 Program 2 Symbol Table
 ---------------------------
 Name  Type     Scope  Line
 ---------------------------
 j     int      0      9
 i     int      0      8


 CODE GENERATION: Beginning Code Generation on Program 2 ...
 CODE GENERATION: Storing value: false in heap at location: 250
 CODE GENERATION: Storing value: true in heap at location: 245
 CODE GENERATION: Adding Variable Declaration of Variable: i
 CODE GENERATION: Adding Variable Declaration of Variable: j
 CODE GENERATION: Assigning Variable i to value: 9
 CODE GENERATION: Assigning Variable j to value: 5
 CODE GENERATION: Printing variable: i
 CODE GENERATION: Printing variable: j
 CODE GENERATION: Adding Break Statement
 CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
21
 CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
22
 Program 2 Code Generation Passed With 0 error(s)

 Program 2 Static Variable Table
 ---------------------------------
 Name  Temp   Address  Scope
 ---------------------------------
 i     T0XX   21        0
 j     T1XX   22        0

 Program 2 Jump Table
 ----------------------
 Temp  Distance
 ----------------------

 Program 2 Machine Code:
 A9 00 8D 21 00 A9 00 8D 22 00 A9 09 8D 21 00 A9
 05 8D 22 00 AC 21 00 A2 01 FF AC 22 00 A2 01 FF
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 74 72 75 65 00 66 61 6C 73 65 00
```

```
INFO  Lexer - Lexing program 3...
DEBUG Lexer - T_L_BRACE [ { ] found at (16:1)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (17:3)
DEBUG Lexer - T_ID [ n ] found at (17:10)
DEBUG Lexer - T_ID [ n ] found at (18:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (18:5)
DEBUG Lexer - T_QUOTE [ " ] found at (18:7)
DEBUG Lexer - T_CHAR [ e ] found at (18:8)
DEBUG Lexer - T_CHAR [ m ] found at (18:9)
DEBUG Lexer - T_CHAR [ i ] found at (18:10)
DEBUG Lexer - T_CHAR [ l ] found at (18:11)
DEBUG Lexer - T_CHAR [ y ] found at (18:12)
DEBUG Lexer - T_QUOTE [ " ] found at (18:13)
DEBUG Lexer - T_PRINT [ print ] found at (19:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (19:8)
DEBUG Lexer - T_ID [ n ] found at (19:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (19:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (20:1)
DEBUG Lexer - T_EOP [ $ ] found at (20:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 3 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 3 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[string]
-----<Id>
------[n]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[n]
------[=]
------<Expression>
-------<StringExpression>
--------["]
--------<CharList>
---------<Char>
----------[e]
---------<CharList>
----------<Char>
-----------[m]
----------<CharList>
-----------<Char>
------------[i]
-----------<CharList>
------------<Char>
-------------[l]
------------<CharList>
-------------<Char>
--------------[y]
--------["]
----<StatementList>
-----<Statement>
------<PrintStatement>
-------[print]
-------[(]
-------<Expression>
--------<Id>
---------[n]
-------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 3 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 16.
SEMANTIC ANALYSIS: Variable [ n ] has been declared at (17:3)
SEMANTIC ANALYSIS: Variable [ n ] has been initialized at (18:7)
SEMANTIC ANALYSIS: Variable [ n ] has been used at (19:9)

Program 3 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 3 ...
<BLOCK>
-<VariableDeclaration>
--[string]
--[n]
-<Assign>
--[n]
--["emily"]
-<Print>
```

```
 --[n]

 Program 3 Symbol Table
 ---------------------------
 Name  Type     Scope  Line
 ---------------------------
 n     string   0      17


 CODE GENERATION: Beginning Code Generation on Program 3 ...
 CODE GENERATION: Storing value: false in heap at location: 250
 CODE GENERATION: Storing value: true in heap at location: 245
 CODE GENERATION: Adding Variable Declaration of Variable: n
 CODE GENERATION: Storing value: emily in heap at location: 239
 CODE GENERATION: Assigning Variable n to value: "emily"
 CODE GENERATION: Printing variable: n
 CODE GENERATION: Adding Break Statement
 CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
11
 Program 3 Code Generation Passed With 0 error(s)

 Program 3 Static Variable Table
 ---------------------------------
 Name  Temp   Address  Scope
 ---------------------------------
 n     T0XX   11        0

 Program 3 Jump Table
 ----------------------
 Temp  Distance
 ----------------------

 Program 3 Machine Code:
 A9 00 8D 11 00 A9 EF 8D 11 00 AC 11 00 A2 02 FF
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 65
 6D 69 6C 79 00 74 72 75 65 00 66 61 6C 73 65 00



 INFO  Lexer - Lexing program 4...
 DEBUG Lexer - T_L_BRACE [ { ] found at (22:1)
 DEBUG Lexer - T_VARIABLE_TYPE [ boolean ] found at (23:3)
 DEBUG Lexer - T_ID [ b ] found at (23:11)
 DEBUG Lexer - T_ID [ b ] found at (24:3)
 DEBUG Lexer - T_ASSIGN_OP [ = ] found at (24:5)
 DEBUG Lexer - T_BOOL_FALSE [ false ] found at (24:7)
 DEBUG Lexer - T_PRINT [ print ] found at (25:3)
 DEBUG Lexer - T_L_PAREN [ ( ] found at (25:8)
 DEBUG Lexer - T_ID [ b ] found at (25:9)
 DEBUG Lexer - T_R_PAREN [ ) ] found at (25:10)
 DEBUG Lexer - T_R_BRACE [ } ] found at (26:1)
 DEBUG Lexer - T_EOP [ $ ] found at (26:2)
 INFO  Lexer - Lex completed with 0 errors

 PARSER: Parsing program 4 ...
 PARSER: parse()
 PARSER: parseProgram()
 PARSER: parseBlock()
 PARSER: parseStatementList()
 PARSER: parseStatement()
 PARSER: parseVarDecl()
 PARSER: parseType()
 PARSER: parseStatementList()
 PARSER: parseStatement()
 PARSER: parseAssignStatement()
 PARSER: parseExpr()
 PARSER: parseBooleanExpr()
 PARSER: parseStatementList()
 PARSER: parseStatement()
 PARSER: parsePrintStatement()
 PARSER: parseExpr()
 PARSER: parseStatementList()
 PARSER: Parse completed successfully

 CST for program 4 ...
 <Program>
 -<Block>
 --[{]
 --<StatementList>
 ---<Statement>
 ----<VarDecl>
 -----<Type>
 ------[boolean]
 -----<Id>
 ------[b]
 ---<StatementList>
 ----<Statement>
 -----<AssignStatement>
 ------<Id>
 -------[b]
 ------[=]
 ------<Expression>
 -------<BooleanExpression>
 --------<BoolVal>
 ---------[false]
 ----<StatementList>
 -----<Statement>
 ------<PrintStatement>
 -------[print]
```

```
-------[(]
-------<Expression>
--------<Id>
---------[b]
-------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 4 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 22.
SEMANTIC ANALYSIS: Variable [ b ] has been declared at (23:3)
SEMANTIC ANALYSIS: Variable [ b ] has been initialized at (24:7)
SEMANTIC ANALYSIS: Variable [ b ] has been used at (25:9)

Program 4 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 4 ...
<BLOCK>
-<VariableDeclaration>
--[boolean]
--[b]
-<Assign>
--[b]
--[false]
-<Print>
--[b]

Program 4 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
b     boolean  0      23


CODE GENERATION: Beginning Code Generation on Program 4 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: b
CODE GENERATION: Assigning Variable b to value: false
CODE GENERATION: Printing variable: b
CODE GENERATION: Adding Break Statement
CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
11
Program 4 Code Generation Passed With 0 error(s)

Program 4 Static Variable Table
---------------------------------
Name  Temp   Address  Scope
---------------------------------
b     T0XX   11        0

Program 4 Jump Table
----------------------
Temp  Distance
----------------------
```

```
Program 4 Machine Code:
A9 FA 8D 11 00 A9 FA 8D 11 00 AC 11 00 A2 02 FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 74 72 75 65 00 66 61 6C 73 65 00
```

The first program creates a variable a, stores the value 1 in it, and outputs it (printing a value of 1). The second program creates two variables, i and j. It sets i to be 9 and j to be 5. Then it prints both of them, so it outputs 9 then 5 (95). The third program declares a string, n, and then sets n = "emily"; this value is stored in the heap. It then prints n which outputs: emily. The fourth program declares a boolean, b, and sets it equal to false. Then, b is printed and false is output.

## Multi-Scope Test Cases (Verbose Mode)

**Input text:**

```
{
  int a
  {
    a = 5
    print(a)
  }
  string x
  x = " hello "
  print(x)
  boolean y
  y = false
  print(y)
}$


{
  string y
  {
    {
      y = "hi"
      print(y)
    }
  }
  string z
  z = "bye"
  {
    y = "bye"
    print(y)
    print(z)
  }
}$

{
  int i
  i = 1
  print(i)
  {
    int i
    {
      print(i)
    }
  }
  print(i)
}$
```

**Output:**

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (2:3)
DEBUG Lexer - T_ID [ a ] found at (2:7)
DEBUG Lexer - T_L_BRACE [ { ] found at (3:3)
DEBUG Lexer - T_ID [ a ] found at (4:5)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (4:7)
DEBUG Lexer - T_DIGIT [ 5 ] found at (4:9)
DEBUG Lexer - T_PRINT [ print ] found at (5:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (5:10)
DEBUG Lexer - T_ID [ a ] found at (5:11)
DEBUG Lexer - T_R_PAREN [ ) ] found at (5:12)
DEBUG Lexer - T_R_BRACE [ } ] found at (6:3)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (7:3)
DEBUG Lexer - T_ID [ x ] found at (7:10)
DEBUG Lexer - T_ID [ x ] found at (8:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (8:5)
DEBUG Lexer - T_QUOTE [ " ] found at (8:7)
DEBUG Lexer - T_CHAR [   ] found at (8:8)
DEBUG Lexer - T_CHAR [ h ] found at (8:9)
DEBUG Lexer - T_CHAR [ e ] found at (8:10)
DEBUG Lexer - T_CHAR [ l ] found at (8:11)
DEBUG Lexer - T_CHAR [ l ] found at (8:12)
DEBUG Lexer - T_CHAR [ o ] found at (8:13)
DEBUG Lexer - T_CHAR [   ] found at (8:14)
DEBUG Lexer - T_QUOTE [ " ] found at (8:15)
DEBUG Lexer - T_PRINT [ print ] found at (9:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (9:8)
DEBUG Lexer - T_ID [ x ] found at (9:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (9:10)
DEBUG Lexer - T_VARIABLE_TYPE [ boolean ] found at (10:3)
DEBUG Lexer - T_ID [ y ] found at (10:11)
DEBUG Lexer - T_ID [ y ] found at (11:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (11:5)
DEBUG Lexer - T_BOOL_FALSE [ false ] found at (11:7)
DEBUG Lexer - T_PRINT [ print ] found at (11:13)
DEBUG Lexer - T_L_PAREN [ ( ] found at (11:18)
DEBUG Lexer - T_ID [ y ] found at (11:19)
DEBUG Lexer - T_R_PAREN [ ) ] found at (11:20)
DEBUG Lexer - T_R_BRACE [ } ] found at (12:1)
DEBUG Lexer - T_EOP [ $ ] found at (12:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 1 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
```

```
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseBooleanExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[a]
---<StatementList>
----<Statement>
-----<Block>
------[{]
------<StatementList>
-------<Statement>
--------<AssignStatement>
---------<Id>
----------[a]
---------[=]
---------<Expression>
----------<IntegerExpression>
-----------<Digit>
------------[5]
-------<StatementList>
--------<Statement>
---------<PrintStatement>
----------[print]
----------[(]
----------<Expression>
-----------<Id>
------------[a]
----------[)]
------[}]
----<StatementList>
-----<Statement>
------<VarDecl>
-------<Type>
--------[string]
-------<Id>
--------[x]
-----<StatementList>
------<Statement>
```

```
-------<AssignStatement>
--------<Id>
---------[x]
--------[=]
--------<Expression>
---------<StringExpression>
----------["]
----------<CharList>
-----------<Char>
------------[ ]
-----------<CharList>
------------<Char>
-------------[h]
------------<CharList>
-------------<Char>
--------------[e]
-------------<CharList>
--------------<Char>
---------------[l]
--------------<CharList>
---------------<Char>
----------------[l]
---------------<CharList>
----------------<Char>
-----------------[o]
----------------<CharList>
-----------------<Char>
------------------[ ]
----------["]
------<StatementList>
-------<Statement>
--------<PrintStatement>
---------[print]
---------[(]
---------<Expression>
----------<Id>
-----------[x]
---------[)]
-------<StatementList>
--------<Statement>
---------<VarDecl>
----------<Type>
-----------[boolean]
----------<Id>
-----------[y]
--------<StatementList>
---------<Statement>
----------<AssignStatement>
-----------<Id>
------------[y]
-----------[=]
-----------<Expression>
------------<BooleanExpression>
-------------<BoolVal>
--------------[false]
---------<StatementList>
```

```
----------<Statement>
-----------<PrintStatement>
------------[print]
------------[(]
------------<Expression>
-------------<Id>
--------------[y]
------------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 1 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 1.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (2:3)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 3.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 3.
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (4:9)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (5:11)
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 6.
SEMANTIC ANALYSIS: Variable [ x ] has been declared at (7:3)
SEMANTIC ANALYSIS: Variable [ x ] has been initialized at (8:7)
SEMANTIC ANALYSIS: Variable [ x ] has been used at (9:9)
SEMANTIC ANALYSIS: Variable [ y ] has been declared at (10:3)
SEMANTIC ANALYSIS: Variable [ y ] has been initialized at (11:7)
SEMANTIC ANALYSIS: Variable [ y ] has been used at (11:19)

Program 1 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 1 ...
<BLOCK>
-<VariableDeclaration>
--[int]
--[a]
-<BLOCK>
--<Assign>
---[a]
---[5]
--<Print>
---[a]
-<VariableDeclaration>
--[string]
--[x]
-<Assign>
--[x]
--[" hello "]
-<Print>
--[x]
-<VariableDeclaration>
--[boolean]
--[y]
-<Assign>
--[y]
--[false]
-<Print>
--[y]
```

```
Program 1 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
x     string   0      7
a     int      0      2
y     boolean  0      10


CODE GENERATION: Beginning Code Generation on Program 1 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: a
CODE GENERATION: Assigning Variable a to value: 5
CODE GENERATION: Printing variable: a
CODE GENERATION: Adding Variable Declaration of Variable: x
CODE GENERATION: Storing value:  hello  in heap at location: 237
CODE GENERATION: Assigning Variable x to value: " hello "
CODE GENERATION: Printing variable: x
CODE GENERATION: Adding Variable Declaration of Variable: y
CODE GENERATION: Assigning Variable y to value: false
CODE GENERATION: Printing variable: y
CODE GENERATION: Adding Break Statement
CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
31
CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
32
CODE GENERATION: Backpatching Static Variable Placeholder T2XX With Memory Address
33
Program 1 Code Generation Passed With 0 error(s)

Program 1 Static Variable Table
--------------------------------
Name  Temp   Address  Scope
--------------------------------
a     T0XX   31        0
x     T1XX   32        0
y     T2XX   33        0

Program 1 Jump Table
---------------------
Temp  Distance
---------------------

Program 1 Machine Code:
A9 00 8D 31 00 A9 05 8D 31 00 AC 31 00 A2 01 FF
A9 00 8D 32 00 A9 ED 8D 32 00 AC 32 00 A2 02 FF
A9 FA 8D 33 00 A9 FA 8D 33 00 AC 33 00 A2 02 FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 20 68 65
6C 6C 6F 20 00 74 72 75 65 00 66 61 6C 73 65 00
```

```
INFO  Lexer - Lexing program 2...
DEBUG Lexer - T_L_BRACE [ { ] found at (15:1)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (16:3)
DEBUG Lexer - T_ID [ y ] found at (16:10)
DEBUG Lexer - T_L_BRACE [ { ] found at (17:3)
DEBUG Lexer - T_L_BRACE [ { ] found at (18:5)
DEBUG Lexer - T_ID [ y ] found at (19:7)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (19:9)
DEBUG Lexer - T_QUOTE [ " ] found at (19:11)
DEBUG Lexer - T_CHAR [ h ] found at (19:12)
DEBUG Lexer - T_CHAR [ i ] found at (19:13)
DEBUG Lexer - T_QUOTE [ " ] found at (19:14)
DEBUG Lexer - T_PRINT [ print ] found at (20:7)
DEBUG Lexer - T_L_PAREN [ ( ] found at (20:12)
DEBUG Lexer - T_ID [ y ] found at (20:13)
DEBUG Lexer - T_R_PAREN [ ) ] found at (20:14)
DEBUG Lexer - T_R_BRACE [ } ] found at (21:5)
DEBUG Lexer - T_R_BRACE [ } ] found at (22:3)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (23:3)
DEBUG Lexer - T_ID [ z ] found at (23:10)
DEBUG Lexer - T_ID [ z ] found at (24:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (24:5)
DEBUG Lexer - T_QUOTE [ " ] found at (24:7)
DEBUG Lexer - T_CHAR [ b ] found at (24:8)
DEBUG Lexer - T_CHAR [ y ] found at (24:9)
DEBUG Lexer - T_CHAR [ e ] found at (24:10)
DEBUG Lexer - T_QUOTE [ " ] found at (24:11)
DEBUG Lexer - T_L_BRACE [ { ] found at (25:3)
DEBUG Lexer - T_ID [ y ] found at (26:5)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (26:7)
DEBUG Lexer - T_QUOTE [ " ] found at (26:9)
DEBUG Lexer - T_CHAR [ b ] found at (26:10)
DEBUG Lexer - T_CHAR [ y ] found at (26:11)
DEBUG Lexer - T_CHAR [ e ] found at (26:12)
DEBUG Lexer - T_QUOTE [ " ] found at (26:13)
DEBUG Lexer - T_PRINT [ print ] found at (27:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (27:10)
DEBUG Lexer - T_ID [ y ] found at (27:11)
DEBUG Lexer - T_R_PAREN [ ) ] found at (27:12)
DEBUG Lexer - T_PRINT [ print ] found at (28:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (28:10)
DEBUG Lexer - T_ID [ z ] found at (28:11)
DEBUG Lexer - T_R_PAREN [ ) ] found at (28:12)
DEBUG Lexer - T_R_BRACE [ } ] found at (29:3)
DEBUG Lexer - T_R_BRACE [ } ] found at (30:1)
DEBUG Lexer - T_EOP [ $ ] found at (30:2)
INFO  Lexer - Lex completed with 0 errors
```

```
PARSER: Parsing program 2 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
```

```
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 2 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[string]
-----<Id>
------[y]
---<StatementList>
----<Statement>
-----<Block>
------[{]
------<StatementList>
-------<Statement>
--------<Block>
---------[{]
---------<StatementList>
----------<Statement>
-----------<AssignStatement>
------------<Id>
-------------[y]
------------[=]
------------<Expression>
-------------<StringExpression>
--------------["]
--------------<CharList>
---------------<Char>
----------------[h]
---------------<CharList>
----------------<Char>
-----------------[i]
--------------["]
----------<StatementList>
-----------<Statement>
------------<PrintStatement>
-------------[print]
-------------[(]
-------------<Expression>
--------------<Id>
---------------[y]
-------------[)]
---------[}]
------[}]
----<StatementList>
-----<Statement>
------<VarDecl>
-------<Type>
```

```
--------[string]
-------<Id>
--------[z]
-----<StatementList>
------<Statement>
-------<AssignStatement>
--------<Id>
---------[z]
--------[=]
--------<Expression>
---------<StringExpression>
----------["]
----------<CharList>
-----------<Char>
------------[b]
-----------<CharList>
------------<Char>
-------------[y]
------------<CharList>
-------------<Char>
--------------[e]
----------["]
------<StatementList>
-------<Statement>
--------<Block>
---------[{]
---------<StatementList>
----------<Statement>
-----------<AssignStatement>
------------<Id>
-------------[y]
------------[=]
------------<Expression>
-------------<StringExpression>
--------------["]
--------------<CharList>
---------------<Char>
----------------[b]
---------------<CharList>
----------------<Char>
-----------------[y]
----------------<CharList>
-----------------<Char>
------------------[e]
--------------["]
----------<StatementList>
-----------<Statement>
------------<PrintStatement>
-------------[print]
-------------[(]
-------------<Expression>
--------------<Id>
---------------[y]
-------------[)]
-----------<StatementList>
------------<Statement>
-------------<PrintStatement>
--------------[print]
--------------[(]
--------------<Expression>
---------------<Id>
----------------[z]
--------------[)]
---------[}]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 2 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 15.
SEMANTIC ANALYSIS: Variable [ y ] has been declared at (16:3)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 17.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 17.
SEMANTIC ANALYSIS: New Scope [ 2 ] has been entered at line: 18.
SEMANTIC ANALYSIS: Scope [ 2 ] parent scope has been set to [ 1 ] at line: 18.
SEMANTIC ANALYSIS: Variable [ y ] has been initialized at (19:11)
SEMANTIC ANALYSIS: Variable [ y ] has been used at (20:13)
SEMANTIC ANALYSIS: Exiting scope [ 2 ] and entering scope [ 1 ] at line: 21.
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 22.
SEMANTIC ANALYSIS: Variable [ z ] has been declared at (23:3)
SEMANTIC ANALYSIS: Variable [ z ] has been initialized at (24:7)
SEMANTIC ANALYSIS: New Scope [ 3 ] has been entered at line: 25.
SEMANTIC ANALYSIS: Scope [ 3 ] parent scope has been set to [ 0 ] at line: 25.
SEMANTIC ANALYSIS: Variable [ y ] has been initialized at (26:9)
SEMANTIC ANALYSIS: Variable [ y ] has been used at (27:11)
SEMANTIC ANALYSIS: Variable [ z ] has been used at (28:11)
SEMANTIC ANALYSIS: Exiting scope [ 3 ] and entering scope [ 0 ] at line: 29.

Program 2 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 2 ...
<BLOCK>
-<VariableDeclaration>
--[string]
--[y]
-<BLOCK>
--<BLOCK>
---<Assign>
----[y]
----["hi"]
---<Print>
----[y]
-<VariableDeclaration>
--[string]
--[z]
-<Assign>
--[z]
--["bye"]
-<BLOCK>
--<Assign>
---[y]
---["bye"]
--<Print>
```

```
---[y]
--<Print>
---[z]

Program 2 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
z     string   0      23
y     string   0      16


CODE GENERATION: Beginning Code Generation on Program 2 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: y
CODE GENERATION: Storing value: hi in heap at location: 242
CODE GENERATION: Assigning Variable y to value: "hi"
CODE GENERATION: Printing variable: y
CODE GENERATION: Adding Variable Declaration of Variable: z
CODE GENERATION: Storing value: bye in heap at location: 238
CODE GENERATION: Assigning Variable z to value: "bye"
CODE GENERATION: Storing value: bye in heap at location: 234
CODE GENERATION: Assigning Variable y to value: "bye"
CODE GENERATION: Printing variable: y
CODE GENERATION: Printing variable: z
CODE GENERATION: Adding Break Statement
CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
2C
CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
2D
Program 2 Code Generation Passed With 0 error(s)

Program 2 Static Variable Table
---------------------------------
Name  Temp   Address  Scope
---------------------------------
y     T0XX   2C        0
z     T1XX   2D        0

Program 2 Jump Table
----------------------
Temp  Distance
----------------------

Program 2 Machine Code:
A9 00 8D 2C 00 A9 F2 8D 2C 00 AC 2C 00 A2 02 FF
A9 00 8D 2D 00 A9 EE 8D 2D 00 A9 EA 8D 2C 00 AC
2C 00 A2 02 FF AC 2D 00 A2 02 FF 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 62 79 65 00 62 79
65 00 68 69 00 74 72 75 65 00 66 61 6C 73 65 00



INFO  Lexer - Lexing program 3...
DEBUG Lexer - T_L_BRACE [ { ] found at (32:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (33:3)
DEBUG Lexer - T_ID [ i ] found at (33:7)
DEBUG Lexer - T_ID [ i ] found at (34:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (34:5)
DEBUG Lexer - T_DIGIT [ 1 ] found at (34:7)
DEBUG Lexer - T_PRINT [ print ] found at (35:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (35:8)
DEBUG Lexer - T_ID [ i ] found at (35:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (35:10)
DEBUG Lexer - T_L_BRACE [ { ] found at (36:3)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (37:5)
DEBUG Lexer - T_ID [ i ] found at (37:9)
DEBUG Lexer - T_L_BRACE [ { ] found at (38:5)
DEBUG Lexer - T_PRINT [ print ] found at (39:7)
DEBUG Lexer - T_L_PAREN [ ( ] found at (39:12)
DEBUG Lexer - T_ID [ i ] found at (39:13)
DEBUG Lexer - T_R_PAREN [ ) ] found at (39:14)
DEBUG Lexer - T_R_BRACE [ } ] found at (40:5)
DEBUG Lexer - T_R_BRACE [ } ] found at (41:3)
DEBUG Lexer - T_PRINT [ print ] found at (42:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (42:8)
DEBUG Lexer - T_ID [ i ] found at (42:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (42:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (43:1)
DEBUG Lexer - T_EOP [ $ ] found at (43:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 3 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
```

```
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 3 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[i]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[i]
------[=]
------<Expression>
-------<IntegerExpression>
--------<Digit>
---------[1]
----<StatementList>
-----<Statement>
------<PrintStatement>
-------[print]
-------[(]
-------<Expression>
--------<Id>
---------[i]
-------[)]
-----<StatementList>
------<Statement>
-------<Block>
--------[{]
--------<StatementList>
---------<Statement>
----------<VarDecl>
-----------<Type>
------------[int]
-----------<Id>
------------[i]
---------<StatementList>
----------<Statement>
-----------<Block>
------------[{]
------------<StatementList>
-------------<Statement>
--------------<PrintStatement>
---------------[print]
---------------[(]
---------------<Expression>
----------------<Id>
-----------------[i]
---------------[)]
------------[}]
--------[}]
------<StatementList>
-------<Statement>
--------<PrintStatement>
---------[print]
---------[(]
---------<Expression>
----------<Id>
-----------[i]
---------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 3 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 32.
SEMANTIC ANALYSIS: Variable [ i ] has been declared at (33:3)
SEMANTIC ANALYSIS: Variable [ i ] has been initialized at (34:7)
SEMANTIC ANALYSIS: Variable [ i ] has been used at (35:9)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 36.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 36.
SEMANTIC ANALYSIS: Variable [ i ] has been declared at (37:5)
SEMANTIC ANALYSIS: New Scope [ 2 ] has been entered at line: 38.
SEMANTIC ANALYSIS: Scope [ 2 ] parent scope has been set to [ 1 ] at line: 38.
SEMANTIC ANALYSIS: Variable [ i ] has been used at (39:13)
SEMANTIC ANALYSIS: Exiting scope [ 2 ] and entering scope [ 1 ] at line: 40.
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 41.
SEMANTIC ANALYSIS: Variable [ i ] has been used at (42:9)
SEMANTIC ANALYSIS: WARNING: Variable [ i ] is declared and used but never
initialized.

Program 3 Semantic Analysis produced 0 error(s) and 1 warning(s).

AST for program 3 ...
<BLOCK>
-<VariableDeclaration>
```

```
--[int]
--[i]
-<Assign>
--[i]
--[1]
-<Print>
--[i]
-<BLOCK>
--<VariableDeclaration>
---[int]
---[i]
--<BLOCK>
---<Print>
----[i]
-<Print>
--[i]

Program 3 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
i     int      0      33
i     int      1      37


CODE GENERATION: Beginning Code Generation on Program 3 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: i
CODE GENERATION: Assigning Variable i to value: 1
CODE GENERATION: Printing variable: i
CODE GENERATION: Adding Variable Declaration of Variable: i
CODE GENERATION: Printing variable: i
CODE GENERATION: Printing variable: i
CODE GENERATION: Adding Break Statement
CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
22
CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
23
Program 3 Code Generation Passed With 0 error(s)

Program 3 Static Variable Table
---------------------------------
Name  Temp   Address  Scope
---------------------------------
i     T0XX   22        0
i     T1XX   23        1

Program 3 Jump Table
----------------------
Temp  Distance
----------------------

Program 3 Machine Code:
A9 00 8D 22 00 A9 01 8D 22 00 AC 22 00 A2 01 FF
A9 00 8D 23 00 AC 23 00 A2 01 FF AC 22 00 A2 01
```

```
FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 74 72 75 65 00 66 61 6C 73 65 00
```

The first program declares an integer a in scope 0. Then, in scope 1, a is set to 5 and printed (5 is output). Scope 1 is exited and a string x is declared and set to " hello " and printed. Then, a boolean y is declared and set to false and printed. The final output of the program is: 5 hello false. In the second program a string y is declared in scope 0. In scope 2, y is set to hi and printed. In scope 0 then string z is declared and set to bye. Finally, in scope 3, y is set to bye and both y and z are printed. The final output of the program is: hibyebye. In the third program, two integers named i are declared in different scopes and output. In scope 0 i is set to 1 and printed. In scope 1 another int i is declared and never initialized (set to default 0). In scope 2 i is printed and 0 is output. Then at the end back in scope 0, i is output again and 1 is output. The final output of the program is 101.

## If Statement Test Cases (Verbose Mode)

**Input text:**

```
{
  string a
  a = "hello"
  string c
  c = a
  if(a == c){
    c = "google"
  }
  print(c)
}$

{
  int a
  a = 5
  if(2==1){
    print(7)
    print("hi")
    print(a)
  }
  print("bye")
}$
```

```
{
  int a
  a = 5
  if (true != false){
    int a
    print("hi")
    a = 2+a
  }
  print(a)
}$
```

**Output:**

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (2:3)
DEBUG Lexer - T_ID [ a ] found at (2:10)
DEBUG Lexer - T_ID [ a ] found at (3:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (3:5)
DEBUG Lexer - T_QUOTE [ " ] found at (3:7)
DEBUG Lexer - T_CHAR [ h ] found at (3:8)
DEBUG Lexer - T_CHAR [ e ] found at (3:9)
DEBUG Lexer - T_CHAR [ l ] found at (3:10)
DEBUG Lexer - T_CHAR [ l ] found at (3:11)
DEBUG Lexer - T_CHAR [ o ] found at (3:12)
DEBUG Lexer - T_QUOTE [ " ] found at (3:13)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (4:3)
DEBUG Lexer - T_ID [ c ] found at (4:10)
DEBUG Lexer - T_ID [ c ] found at (5:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (5:5)
DEBUG Lexer - T_ID [ a ] found at (5:7)
DEBUG Lexer - T_IF [ if ] found at (6:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (6:5)
DEBUG Lexer - T_ID [ a ] found at (6:6)
DEBUG Lexer - T_EQUALITY_OP [ == ] found at (6:8)
DEBUG Lexer - T_ID [ c ] found at (6:11)
DEBUG Lexer - T_R_PAREN [ ) ] found at (6:12)
DEBUG Lexer - T_L_BRACE [ { ] found at (6:13)
DEBUG Lexer - T_ID [ c ] found at (7:5)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (7:7)
DEBUG Lexer - T_QUOTE [ " ] found at (7:9)
DEBUG Lexer - T_CHAR [ g ] found at (7:10)
DEBUG Lexer - T_CHAR [ o ] found at (7:11)
DEBUG Lexer - T_CHAR [ o ] found at (7:12)
DEBUG Lexer - T_CHAR [ g ] found at (7:13)
DEBUG Lexer - T_CHAR [ l ] found at (7:14)
DEBUG Lexer - T_CHAR [ e ] found at (7:15)
DEBUG Lexer - T_QUOTE [ " ] found at (7:16)
DEBUG Lexer - T_R_BRACE [ } ] found at (8:3)
DEBUG Lexer - T_PRINT [ print ] found at (9:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (9:8)
DEBUG Lexer - T_ID [ c ] found at (9:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (9:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (10:1)
DEBUG Lexer - T_EOP [ $ ] found at (10:2)
INFO  Lexer - Lex completed with 0 errors
```

```
PARSER: Parsing program 1 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseIfStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseBoolOp()
PARSER: parseExpr()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1 ...
<Program>
```

```
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[string]
-----<Id>
------[a]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[a]
------[=]
------<Expression>
-------<StringExpression>
--------["]
--------<CharList>
---------<Char>
----------[h]
---------<CharList>
----------<Char>
-----------[e]
----------<CharList>
-----------<Char>
------------[l]
-----------<CharList>
------------<Char>
-------------[l]
------------<CharList>
-------------<Char>
--------------[o]
--------["]
----<StatementList>
-----<Statement>
------<VarDecl>
-------<Type>
--------[string]
-------<Id>
--------[c]
-----<StatementList>
------<Statement>
-------<AssignStatement>
--------<Id>
---------[c]
--------[=]
--------<Expression>
---------<Id>
----------[a]
------<StatementList>
-------<Statement>
--------<IfStatement>
---------[if]
---------<BooleanExpression>
----------[(]
----------<Expression>
-----------<Id>
------------[a]
----------<BoolOp>
-----------[==]
----------<Expression>
-----------<Id>
------------[c]
----------[)]
---------<Block>
----------[{]
----------<StatementList>
-----------<Statement>
------------<AssignStatement>
-------------<Id>
--------------[c]
-------------[=]
-------------<Expression>
--------------<StringExpression>
---------------["]
---------------<CharList>
----------------<Char>
-----------------[g]
----------------<CharList>
-----------------<Char>
------------------[o]
-----------------<CharList>
------------------<Char>
-------------------[o]
------------------<CharList>
-------------------<Char>
--------------------[g]
-------------------<CharList>
--------------------<Char>
---------------------[l]
--------------------<CharList>
---------------------<Char>
----------------------[e]
---------------["]
----------[}]
-------<StatementList>
--------<Statement>
---------<PrintStatement>
----------[print]
----------[(]
----------<Expression>
-----------<Id>
------------[c]
----------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 1 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 1.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (2:3)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (3:7)
SEMANTIC ANALYSIS: Variable [ c ] has been declared at (4:3)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (5:7)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (6:6)
SEMANTIC ANALYSIS: Variable [ c ] has been used at (6:11)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 6.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 6.
SEMANTIC ANALYSIS: Variable [ c ] has been initialized at (7:9)
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 8.
SEMANTIC ANALYSIS: Variable [ c ] has been used at (9:9)

Program 1 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 1 ...
<BLOCK>
-<VariableDeclaration>
--[string]
--[a]
-<Assign>
--[a]
--["hello"]
-<VariableDeclaration>
--[string]
--[c]
-<Assign>
--[c]
--[a]
-<If>
--<isEqual>
---[a]
---[c]
--<BLOCK>
---<Assign>
----[c]
----["google"]
-<Print>
--[c]

Program 1 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
a     string   0      2
c     string   0      4


CODE GENERATION: Beginning Code Generation on Program 1 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: a
CODE GENERATION: Storing value: hello in heap at location: 239
CODE GENERATION: Assigning Variable a to value: "hello"
CODE GENERATION: Adding Variable Declaration of Variable: c
CODE GENERATION: Assigning Variable c to variable: a
CODE GENERATION: Comparing values: a and c in equality operation.
CODE GENERATION: Storing value: google in heap at location: 232
CODE GENERATION: Assigning Variable c to value: "google"
```

```
 CODE GENERATION: Printing variable: c
 CODE GENERATION: Adding Break Statement
 CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
3F
 CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
40
 CODE GENERATION: Backpatching Static Variable Placeholder T2XX With Memory Address
41
 CODE GENERATION: Backpatching Jump Variable Placeholder J0 Forward 0A Addresses
 Program 1 Code Generation Passed With 0 error(s)

 Program 1 Static Variable Table
 ---------------------------------
 Name  Temp   Address  Scope
 ---------------------------------
 a     T0XX   3F        0
 c     T1XX   40        0
 0     T2XX   41       -1

 Program 1 Jump Table
 ----------------------
 Temp  Distance
 ----------------------
 J0    A

 Program 1 Machine Code:
 A9 00 8D 3F 00 A9 EF 8D 3F 00 A9 00 8D 40 00 AD
 3F 00 8D 40 00 AE 3F 00 EC 40 00 A9 FA 8D 41 00
 D0 05 A9 F5 8D 41 00 A2 F5 EC 41 00 D0 0A A9 FA
 8D 41 00 A9 E8 8D 40 00 AC 40 00 A2 02 FF 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 67 6F 6F 67 6C 65 00 68
 65 6C 6C 6F 00 74 72 75 65 00 66 61 6C 73 65 00



 INFO  Lexer - Lexing program 2...
 DEBUG Lexer - T_L_BRACE [ { ] found at (12:1)
 DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (13:3)
 DEBUG Lexer - T_ID [ a ] found at (13:7)
 DEBUG Lexer - T_ID [ a ] found at (14:3)
 DEBUG Lexer - T_ASSIGN_OP [ = ] found at (14:5)
 DEBUG Lexer - T_DIGIT [ 5 ] found at (14:7)
 DEBUG Lexer - T_IF [ if ] found at (15:3)
 DEBUG Lexer - T_L_PAREN [ ( ] found at (15:5)
 DEBUG Lexer - T_DIGIT [ 2 ] found at (15:6)
 DEBUG Lexer - T_EQUALITY_OP [ == ] found at (15:7)
 DEBUG Lexer - T_DIGIT [ 1 ] found at (15:9)
 DEBUG Lexer - T_R_PAREN [ ) ] found at (15:10)
 DEBUG Lexer - T_L_BRACE [ { ] found at (15:11)
 DEBUG Lexer - T_PRINT [ print ] found at (16:5)
 DEBUG Lexer - T_L_PAREN [ ( ] found at (16:10)
 DEBUG Lexer - T_DIGIT [ 7 ] found at (16:11)
 DEBUG Lexer - T_R_PAREN [ ) ] found at (16:12)
 DEBUG Lexer - T_PRINT [ print ] found at (17:5)
 DEBUG Lexer - T_L_PAREN [ ( ] found at (17:10)
 DEBUG Lexer - T_QUOTE [ " ] found at (17:11)
 DEBUG Lexer - T_CHAR [ h ] found at (17:12)
 DEBUG Lexer - T_CHAR [ i ] found at (17:13)
 DEBUG Lexer - T_QUOTE [ " ] found at (17:14)
 DEBUG Lexer - T_R_PAREN [ ) ] found at (17:15)
 DEBUG Lexer - T_PRINT [ print ] found at (18:5)
 DEBUG Lexer - T_L_PAREN [ ( ] found at (18:10)
 DEBUG Lexer - T_ID [ a ] found at (18:11)
 DEBUG Lexer - T_R_PAREN [ ) ] found at (18:12)
 DEBUG Lexer - T_R_BRACE [ } ] found at (19:3)
 DEBUG Lexer - T_PRINT [ print ] found at (20:3)
 DEBUG Lexer - T_L_PAREN [ ( ] found at (20:8)
 DEBUG Lexer - T_QUOTE [ " ] found at (20:9)
 DEBUG Lexer - T_CHAR [ b ] found at (20:10)
 DEBUG Lexer - T_CHAR [ y ] found at (20:11)
 DEBUG Lexer - T_CHAR [ e ] found at (20:12)
 DEBUG Lexer - T_QUOTE [ " ] found at (20:13)
 DEBUG Lexer - T_R_PAREN [ ) ] found at (20:14)
 DEBUG Lexer - T_R_BRACE [ } ] found at (21:1)
 DEBUG Lexer - T_EOP [ $ ] found at (21:2)
 INFO  Lexer - Lex completed with 0 errors

 PARSER: Parsing program 2 ...
 PARSER: parse()
 PARSER: parseProgram()
 PARSER: parseBlock()
 PARSER: parseStatementList()
 PARSER: parseStatement()
 PARSER: parseVarDecl()
 PARSER: parseType()
 PARSER: parseStatementList()
 PARSER: parseStatement()
 PARSER: parseAssignStatement()
 PARSER: parseExpr()
 PARSER: parseIntExpr()
 PARSER: parseStatementList()
 PARSER: parseStatement()
 PARSER: parseIfStatement()
 PARSER: parseBooleanExpr()
 PARSER: parseExpr()
 PARSER: parseIntExpr()
 PARSER: parseBoolOp()
 PARSER: parseExpr()
 PARSER: parseIntExpr()
 PARSER: parseBlock()
 PARSER: parseStatementList()
 PARSER: parseStatement()
```

```
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 2 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[a]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[a]
------[=]
------<Expression>
-------<IntegerExpression>
--------<Digit>
---------[5]
----<StatementList>
-----<Statement>
------<IfStatement>
-------[if]
-------<BooleanExpression>
--------[(]
--------<Expression>
---------<IntegerExpression>
----------<Digit>
```

```
-----------[2]
--------<BoolOp>
---------[==]
--------<Expression>
---------<IntegerExpression>
----------<Digit>
-----------[1]
--------[)]
-------<Block>
--------[{]
--------<StatementList>
---------<Statement>
----------<PrintStatement>
-----------[print]
-----------[(]
-----------<Expression>
------------<IntegerExpression>
-------------<Digit>
--------------[7]
-----------[)]
---------<StatementList>
----------<Statement>
-----------<PrintStatement>
------------[print]
------------[(]
------------<Expression>
-------------<StringExpression>
--------------["]
--------------<CharList>
---------------<Char>
----------------[h]
---------------<CharList>
----------------<Char>
-----------------[i]
--------------["]
------------[)]
----------<StatementList>
-----------<Statement>
------------<PrintStatement>
-------------[print]
-------------[(]
-------------<Expression>
--------------<Id>
---------------[a]
-------------[)]
--------[}]
-----<StatementList>
------<Statement>
-------<PrintStatement>
--------[print]
--------[(]
--------<Expression>
---------<StringExpression>
----------["]
----------<CharList>
-----------<Char>
```

```
------------[b]
-----------<CharList>
------------<Char>
-------------[y]
------------<CharList>
-------------<Char>
--------------[e]
----------["]
--------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 2 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 12.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (13:3)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (14:7)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 15.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 15.
SEMANTIC ANALYSIS: Variable [ a ] has been used at (18:11)
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 19.

Program 2 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 2 ...
<BLOCK>
-<VariableDeclaration>
--[int]
--[a]
-<Assign>
--[a]
--[5]
-<If>
--<isEqual>
---[2]
---[1]
--<BLOCK>
---<Print>
----[7]
---<Print>
----["hi"]
---<Print>
----[a]
-<Print>
--["bye"]

Program 2 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
a     int      0      13


CODE GENERATION: Beginning Code Generation on Program 2 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: a
```

```
 CODE GENERATION: Assigning Variable a to value: 5
 CODE GENERATION: Comparing values: 2 and 1 in equality operation.
 CODE GENERATION: Printing value: 7
 CODE GENERATION: Storing value: hi in heap at location: 242
 CODE GENERATION: Printing value: "hi"
 CODE GENERATION: Printing variable: a
 CODE GENERATION: Storing value: bye in heap at location: 238
 CODE GENERATION: Printing value: "bye"
 CODE GENERATION: Adding Break Statement
 CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
48
 CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
49
 CODE GENERATION: Backpatching Static Variable Placeholder T2XX With Memory Address
4A
 CODE GENERATION: Backpatching Static Variable Placeholder T3XX With Memory Address
4B
 CODE GENERATION: Backpatching Jump Variable Placeholder J0 Forward 15 Addresses
 Program 2 Code Generation Passed With 0 error(s)

 Program 2 Static Variable Table
 ---------------------------------
 Name  Temp   Address  Scope
 ---------------------------------
 a     T0XX   48        0
 0     T1XX   49       -1
 1     T2XX   4A       -1
 2     T3XX   4B       -1

 Program 2 Jump Table
 ----------------------
 Temp  Distance
 ----------------------
 J0    15

 Program 2 Machine Code:
 A9 00 8D 48 00 A9 05 8D 48 00 A9 02 8D 49 00 A9
 01 8D 4A 00 AE 49 00 EC 4A 00 A9 FA 8D 4B 00 D0
 05 A9 F5 8D 4B 00 A2 F5 EC 4B 00 D0 15 A9 FA 8D
 4B 00 A0 07 A2 01 FF A0 F2 A2 02 FF AC 48 00 A2
 01 FF A0 EE A2 02 FF 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 62 79
 65 00 68 69 00 74 72 75 65 00 66 61 6C 73 65 00


 INFO  Lexer - Lexing program 3...
```

```
DEBUG Lexer - T_L_BRACE [ { ] found at (24:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (25:3)
DEBUG Lexer - T_ID [ a ] found at (25:7)
DEBUG Lexer - T_ID [ a ] found at (26:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (26:5)
DEBUG Lexer - T_DIGIT [ 5 ] found at (26:7)
DEBUG Lexer - T_IF [ if ] found at (27:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (27:6)
DEBUG Lexer - T_BOOL_TRUE [ true ] found at (27:7)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (27:12)
DEBUG Lexer - T_BOOL_FALSE [ false ] found at (27:15)
DEBUG Lexer - T_R_PAREN [ ) ] found at (27:20)
DEBUG Lexer - T_L_BRACE [ { ] found at (27:21)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (28:5)
DEBUG Lexer - T_ID [ a ] found at (28:9)
DEBUG Lexer - T_PRINT [ print ] found at (29:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (29:10)
DEBUG Lexer - T_QUOTE [ " ] found at (29:11)
DEBUG Lexer - T_CHAR [ h ] found at (29:12)
DEBUG Lexer - T_CHAR [ i ] found at (29:13)
DEBUG Lexer - T_QUOTE [ " ] found at (29:14)
DEBUG Lexer - T_R_PAREN [ ) ] found at (29:15)
DEBUG Lexer - T_ID [ a ] found at (30:5)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (30:7)
DEBUG Lexer - T_DIGIT [ 2 ] found at (30:9)
DEBUG Lexer - T_ADDITION_OP [ + ] found at (30:10)
DEBUG Lexer - T_ID [ a ] found at (30:11)
DEBUG Lexer - T_R_BRACE [ } ] found at (31:3)
DEBUG Lexer - T_PRINT [ print ] found at (32:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (32:8)
DEBUG Lexer - T_ID [ a ] found at (32:9)
DEBUG Lexer - T_R_PAREN [ ) ] found at (32:10)
DEBUG Lexer - T_R_BRACE [ } ] found at (33:1)
DEBUG Lexer - T_EOP [ $ ] found at (33:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 3 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseIfStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseBooleanExpr()
PARSER: parseBoolOp()
```

```
PARSER: parseExpr()
PARSER: parseBooleanExpr()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 3 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[a]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[a]
------[=]
------<Expression>
-------<IntegerExpression>
--------<Digit>
---------[5]
----<StatementList>
-----<Statement>
------<IfStatement>
-------[if]
-------<BooleanExpression>
--------[(]
--------<Expression>
```

```
---------<BooleanExpression>
----------<BoolVal>
-----------[true]
--------<BoolOp>
---------[!=]
--------<Expression>
---------<BooleanExpression>
----------<BoolVal>
-----------[false]
--------[)]
-------<Block>
--------[{]
--------<StatementList>
---------<Statement>
----------<VarDecl>
-----------<Type>
------------[int]
-----------<Id>
------------[a]
---------<StatementList>
----------<Statement>
-----------<PrintStatement>
------------[print]
------------[(]
------------<Expression>
-------------<StringExpression>
--------------["]
--------------<CharList>
---------------<Char>
----------------[h]
---------------<CharList>
----------------<Char>
-----------------[i]
--------------["]
------------[)]
----------<StatementList>
-----------<Statement>
------------<AssignStatement>
-------------<Id>
--------------[a]
-------------[=]
-------------<Expression>
--------------<IntegerExpression>
---------------<Digit>
----------------[2]
---------------<IntOp>
----------------[+]
---------------<Expression>
----------------<Id>
-----------------[a]
--------[}]
-----<StatementList>
------<Statement>
-------<PrintStatement>
--------[print]
--------[(]
--------<Expression>
---------<Id>
----------[a]
--------[)]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 3 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 24.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (25:3)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (26:7)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 27.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 27.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (28:5)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (30:11)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (30:11)
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 31.
SEMANTIC ANALYSIS: Variable [ a ] has been used at (32:9)

Program 3 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 3 ...
<BLOCK>
-<VariableDeclaration>
--[int]
--[a]
-<Assign>
--[a]
--[5]
-<If>
--<isNotEqual>
---[true]
---[false]
--<BLOCK>
---<VariableDeclaration>
----[int]
----[a]
---<Print>
----["hi"]
---<Assign>
----[a]
----<Addition>
-----[2]
-----[a]
-<Print>
--[a]

Program 3 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
a     int      0      25
a     int      1      28


CODE GENERATION: Beginning Code Generation on Program 3 ...
```

```
 CODE GENERATION: Storing value: false in heap at location: 250
 CODE GENERATION: Storing value: true in heap at location: 245
 CODE GENERATION: Adding Variable Declaration of Variable: a
 CODE GENERATION: Assigning Variable a to value: 5
 CODE GENERATION: Comparing values: true and false in inequality operation.
 CODE GENERATION: Adding Variable Declaration of Variable: a
 CODE GENERATION: Storing value: hi in heap at location: 242
 CODE GENERATION: Printing value: "hi"
 CODE GENERATION: Storing Addition Operation: 2 + a in variable: a
 CODE GENERATION: Printing variable: a
 CODE GENERATION: Adding Break Statement
 CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
4F
 CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
50
 CODE GENERATION: Backpatching Static Variable Placeholder T2XX With Memory Address
51
 CODE GENERATION: Backpatching Static Variable Placeholder T3XX With Memory Address
52
 CODE GENERATION: Backpatching Static Variable Placeholder T4XX With Memory Address
53
 CODE GENERATION: Backpatching Jump Variable Placeholder J0 Forward 25 Addresses
 Program 3 Code Generation Passed With 0 error(s)

 Program 3 Static Variable Table
 --------------------------------
 Name  Temp   Address  Scope
 --------------------------------
 a     T0XX   4F        0
 0     T1XX   50       -1
 a     T2XX   51        1
 1     T3XX   52       -1
 2     T4XX   53       -1

 Program 3 Jump Table
 ----------------------
 Temp  Distance
 ----------------------
 J0    25

 Program 3 Machine Code:
 A9 00 8D 4F 00 A9 05 8D 4F 00 AE F5 00 EC FA 00
 A9 F5 8D 50 00 D0 05 A9 FA 8D 50 00 A2 F5 EC 50
 00 D0 25 A9 F5 8D 50 00 A9 00 8D 51 00 A0 F2 A2
 02 FF A9 02 8D 52 00 A9 00 6D 51 00 6D 52 00 8D
 53 00 AD 53 00 8D 51 00 AC 4F 00 A2 01 FF 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00 00 68 69 00 74 72 75 65 00 66 61 6C 73 65 00
```

The first program creates a string, and sets it equal to hello. Then, another string, c, is declared and set equal to a (value of hello). The if checks if a == c (which is does), so it enters the if and sets c to google. Then, outside the loop, c is printed. The final output of the program is: google. The second program declares int a and sets a equal to 5. The if checks if 2==1 (which it never will be). So it jumps to the end of the if and just outputs: bye. The third program declares an int a and sets it equal to 5 in scope 0. Then the if checks if true!=false which they are not equal, so we enter the if statement. In scope 1 another int a is declared and set to 2+a (2+0). The string "hi" is output within the if also. Then, a is printed in scope 0. The final output of the program is: hi5.

## While Statement Test Cases (Verbose Mode)

**Input text:**

```
{
  int a
  a = 1
  while (a != 1)
  {
    a = 1+a
    print(a)
  }

  {
    boolean b
    b = true
    while (b != false)
    {
      b = false
      string a
      a = "a"
      print(a)
    }
    while (a != 3)
    {
      print(a)
      a = 1+a
    }

    print("bye")
  }
}$
```

**Output:**

```
INFO  Lexer - Lexing program 1...
DEBUG Lexer - T_L_BRACE [ { ] found at (1:1)
DEBUG Lexer - T_VARIABLE_TYPE [ int ] found at (2:3)
DEBUG Lexer - T_ID [ a ] found at (2:7)
DEBUG Lexer - T_ID [ a ] found at (3:3)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (3:5)
DEBUG Lexer - T_DIGIT [ 1 ] found at (3:7)
DEBUG Lexer - T_WHILE [ while ] found at (4:3)
DEBUG Lexer - T_L_PAREN [ ( ] found at (4:9)
DEBUG Lexer - T_ID [ a ] found at (4:10)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (4:12)
DEBUG Lexer - T_DIGIT [ 1 ] found at (4:15)
DEBUG Lexer - T_R_PAREN [ ) ] found at (4:16)
DEBUG Lexer - T_L_BRACE [ { ] found at (5:3)
DEBUG Lexer - T_ID [ a ] found at (6:5)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (6:7)
DEBUG Lexer - T_DIGIT [ 1 ] found at (6:9)
DEBUG Lexer - T_ADDITION_OP [ + ] found at (6:10)
DEBUG Lexer - T_ID [ a ] found at (6:11)
DEBUG Lexer - T_PRINT [ print ] found at (7:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (7:10)
DEBUG Lexer - T_ID [ a ] found at (7:11)
DEBUG Lexer - T_R_PAREN [ ) ] found at (7:12)
DEBUG Lexer - T_R_BRACE [ } ] found at (8:3)
DEBUG Lexer - T_L_BRACE [ { ] found at (10:3)
DEBUG Lexer - T_VARIABLE_TYPE [ boolean ] found at (11:5)
DEBUG Lexer - T_ID [ b ] found at (11:13)
DEBUG Lexer - T_ID [ b ] found at (12:5)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (12:7)
DEBUG Lexer - T_BOOL_TRUE [ true ] found at (12:9)
DEBUG Lexer - T_WHILE [ while ] found at (13:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (13:11)
DEBUG Lexer - T_ID [ b ] found at (13:12)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (13:14)
DEBUG Lexer - T_BOOL_FALSE [ false ] found at (13:17)
DEBUG Lexer - T_R_PAREN [ ) ] found at (13:22)
DEBUG Lexer - T_L_BRACE [ { ] found at (14:5)
DEBUG Lexer - T_ID [ b ] found at (15:7)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (15:9)
DEBUG Lexer - T_BOOL_FALSE [ false ] found at (15:11)
DEBUG Lexer - T_VARIABLE_TYPE [ string ] found at (16:7)
DEBUG Lexer - T_ID [ a ] found at (16:14)
DEBUG Lexer - T_ID [ a ] found at (17:7)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (17:9)
DEBUG Lexer - T_QUOTE [ " ] found at (17:11)
DEBUG Lexer - T_CHAR [ a ] found at (17:12)
DEBUG Lexer - T_QUOTE [ " ] found at (17:13)
DEBUG Lexer - T_PRINT [ print ] found at (18:7)
DEBUG Lexer - T_L_PAREN [ ( ] found at (18:12)
DEBUG Lexer - T_ID [ a ] found at (18:13)
DEBUG Lexer - T_R_PAREN [ ) ] found at (18:14)
DEBUG Lexer - T_R_BRACE [ } ] found at (19:5)
DEBUG Lexer - T_WHILE [ while ] found at (20:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (20:11)
DEBUG Lexer - T_ID [ a ] found at (20:12)
DEBUG Lexer - T_INEQUALITY_OP [ != ] found at (20:14)
DEBUG Lexer - T_DIGIT [ 3 ] found at (20:17)
DEBUG Lexer - T_R_PAREN [ ) ] found at (20:18)
DEBUG Lexer - T_L_BRACE [ { ] found at (21:5)
DEBUG Lexer - T_PRINT [ print ] found at (22:7)
DEBUG Lexer - T_L_PAREN [ ( ] found at (22:12)
DEBUG Lexer - T_ID [ a ] found at (22:13)
DEBUG Lexer - T_R_PAREN [ ) ] found at (22:14)
DEBUG Lexer - T_ID [ a ] found at (23:7)
DEBUG Lexer - T_ASSIGN_OP [ = ] found at (23:9)
DEBUG Lexer - T_DIGIT [ 1 ] found at (23:11)
DEBUG Lexer - T_ADDITION_OP [ + ] found at (23:12)
DEBUG Lexer - T_ID [ a ] found at (23:13)
DEBUG Lexer - T_R_BRACE [ } ] found at (24:5)
DEBUG Lexer - T_PRINT [ print ] found at (26:5)
DEBUG Lexer - T_L_PAREN [ ( ] found at (26:10)
DEBUG Lexer - T_QUOTE [ " ] found at (26:11)
DEBUG Lexer - T_CHAR [ b ] found at (26:12)
DEBUG Lexer - T_CHAR [ y ] found at (26:13)
DEBUG Lexer - T_CHAR [ e ] found at (26:14)
DEBUG Lexer - T_QUOTE [ " ] found at (26:15)
DEBUG Lexer - T_R_PAREN [ ) ] found at (26:16)
DEBUG Lexer - T_R_BRACE [ } ] found at (27:3)
DEBUG Lexer - T_R_BRACE [ } ] found at (28:1)
DEBUG Lexer - T_EOP [ $ ] found at (28:2)
INFO  Lexer - Lex completed with 0 errors

PARSER: Parsing program 1 ...
PARSER: parse()
PARSER: parseProgram()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseWhileStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseBoolOp()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseBlock()
PARSER: parseStatementList()
```

```
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseBooleanExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseWhileStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseBoolOp()
PARSER: parseExpr()
PARSER: parseBooleanExpr()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseBooleanExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseVarDecl()
PARSER: parseType()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseWhileStatement()
PARSER: parseBooleanExpr()
PARSER: parseExpr()
PARSER: parseBoolOp()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseBlock()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parseAssignStatement()
PARSER: parseExpr()
PARSER: parseIntExpr()
PARSER: parseExpr()
```

```
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: parseStatement()
PARSER: parsePrintStatement()
PARSER: parseExpr()
PARSER: parseStringExpr()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseCharList()
PARSER: parseStatementList()
PARSER: parseStatementList()
PARSER: Parse completed successfully

CST for program 1 ...
<Program>
-<Block>
--[{]
--<StatementList>
---<Statement>
----<VarDecl>
-----<Type>
------[int]
-----<Id>
------[a]
---<StatementList>
----<Statement>
-----<AssignStatement>
------<Id>
-------[a]
------[=]
------<Expression>
-------<IntegerExpression>
--------<Digit>
---------[1]
----<StatementList>
-----<Statement>
------<WhileStatement>
-------[while]
-------<BooleanExpression>
--------[(]
--------<Expression>
---------<Id>
----------[a]
--------<BoolOp>
---------[!=]
--------<Expression>
---------<IntegerExpression>
----------<Digit>
-----------[1]
--------[)]
-------<Block>
--------[{]
--------<StatementList>
---------<Statement>
----------<AssignStatement>
-----------<Id>
```

```
------------[a]
-----------[=]
-----------<Expression>
------------<IntegerExpression>
-------------<Digit>
--------------[1]
-------------<IntOp>
--------------[+]
-------------<Expression>
--------------<Id>
---------------[a]
---------<StatementList>
----------<Statement>
-----------<PrintStatement>
------------[print]
------------[(]
------------<Expression>
-------------<Id>
--------------[a]
------------[)]
--------[}]
-----<StatementList>
------<Statement>
-------<Block>
--------[{]
--------<StatementList>
---------<Statement>
----------<VarDecl>
-----------<Type>
------------[boolean]
-----------<Id>
------------[b]
---------<StatementList>
----------<Statement>
-----------<AssignStatement>
------------<Id>
-------------[b]
------------[=]
------------<Expression>
-------------<BooleanExpression>
--------------<BoolVal>
---------------[true]
----------<StatementList>
-----------<Statement>
------------<WhileStatement>
-------------[while]
-------------<BooleanExpression>
--------------[(]
--------------<Expression>
---------------<Id>
----------------[b]
--------------<BoolOp>
---------------[!=]
--------------<Expression>
---------------<BooleanExpression>
----------------<BoolVal>
```

```
-----------------[false]
--------------[)]
-------------<Block>
--------------[{]
--------------<StatementList>
---------------<Statement>
----------------<AssignStatement>
-----------------<Id>
------------------[b]
-----------------[=]
-----------------<Expression>
------------------<BooleanExpression>
-------------------<BoolVal>
--------------------[false]
---------------<StatementList>
----------------<Statement>
-----------------<VarDecl>
------------------<Type>
-------------------[string]
------------------<Id>
-------------------[a]
----------------<StatementList>
-----------------<Statement>
------------------<AssignStatement>
-------------------<Id>
--------------------[a]
-------------------[=]
-------------------<Expression>
--------------------<StringExpression>
---------------------["]
---------------------<CharList>
----------------------<Char>
-----------------------[a]
---------------------["]
-----------------<StatementList>
------------------<Statement>
-------------------<PrintStatement>
--------------------[print]
--------------------[(]
--------------------<Expression>
---------------------<Id>
----------------------[a]
--------------------[)]
--------------[}]
-----------<StatementList>
------------<Statement>
-------------<WhileStatement>
--------------[while]
--------------<BooleanExpression>
---------------[(]
---------------<Expression>
----------------<Id>
-----------------[a]
---------------<BoolOp>
----------------[!=]
---------------<Expression>
```

```
----------------<IntegerExpression>
-----------------<Digit>
------------------[3]
---------------[)]
--------------<Block>
---------------[{]
---------------<StatementList>
----------------<Statement>
-----------------<PrintStatement>
------------------[print]
------------------[(]
------------------<Expression>
-------------------<Id>
--------------------[a]
------------------[)]
----------------<StatementList>
-----------------<Statement>
------------------<AssignStatement>
-------------------<Id>
--------------------[a]
-------------------[=]
-------------------<Expression>
--------------------<IntegerExpression>
---------------------<Digit>
----------------------[1]
---------------------<IntOp>
----------------------[+]
---------------------<Expression>
----------------------<Id>
-----------------------[a]
---------------[}]
------------<StatementList>
-------------<Statement>
--------------<PrintStatement>
---------------[print]
---------------[(]
---------------<Expression>
----------------<StringExpression>
-----------------["]
-----------------<CharList>
------------------<Char>
-------------------[b]
------------------<CharList>
-------------------<Char>
--------------------[y]
-------------------<CharList>
--------------------<Char>
---------------------[e]
-----------------["]
---------------[)]
--------[}]
--[}]
-[$]

SEMANTIC ANALYSIS: Beginning Semantic Analysis on Program 1 ...
SEMANTIC ANALYSIS: New Scope [ 0 ] has been entered at line: 1.
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (2:3)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (3:7)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (4:10)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (4:15)
SEMANTIC ANALYSIS: New Scope [ 1 ] has been entered at line: 5.
SEMANTIC ANALYSIS: Scope [ 1 ] parent scope has been set to [ 0 ] at line: 5.
SEMANTIC ANALYSIS: Variable [ a ] has been used at (6:11)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (6:11)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (7:11)
SEMANTIC ANALYSIS: Exiting scope [ 1 ] and entering scope [ 0 ] at line: 8.
SEMANTIC ANALYSIS: New Scope [ 2 ] has been entered at line: 10.
SEMANTIC ANALYSIS: Scope [ 2 ] parent scope has been set to [ 0 ] at line: 10.
SEMANTIC ANALYSIS: Variable [ b ] has been declared at (11:5)
SEMANTIC ANALYSIS: Variable [ b ] has been initialized at (12:9)
SEMANTIC ANALYSIS: Variable [ b ] has been used at (13:12)
SEMANTIC ANALYSIS: New Scope [ 3 ] has been entered at line: 14.
SEMANTIC ANALYSIS: Scope [ 3 ] parent scope has been set to [ 2 ] at line: 14.
SEMANTIC ANALYSIS: Variable [ b ] has been initialized at (15:11)
SEMANTIC ANALYSIS: Variable [ a ] has been declared at (16:7)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (17:11)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (18:13)
SEMANTIC ANALYSIS: Exiting scope [ 3 ] and entering scope [ 2 ] at line: 19.
SEMANTIC ANALYSIS: Variable [ a ] has been used at (20:12)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (20:17)
SEMANTIC ANALYSIS: New Scope [ 4 ] has been entered at line: 21.
SEMANTIC ANALYSIS: Scope [ 4 ] parent scope has been set to [ 2 ] at line: 21.
SEMANTIC ANALYSIS: Variable [ a ] has been used at (22:13)
SEMANTIC ANALYSIS: Variable [ a ] has been used at (23:13)
SEMANTIC ANALYSIS: Variable [ a ] has been initialized at (23:13)
SEMANTIC ANALYSIS: Exiting scope [ 4 ] and entering scope [ 2 ] at line: 24.
SEMANTIC ANALYSIS: Exiting scope [ 2 ] and entering scope [ 0 ] at line: 27.

Program 1 Semantic Analysis produced 0 error(s) and 0 warning(s).

AST for program 1 ...
<BLOCK>
-<VariableDeclaration>
--[int]
--[a]
-<Assign>
--[a]
--[1]
-<While>
--<isNotEqual>
---[a]
---[1]
--<BLOCK>
---<Assign>
----[a]
----<Addition>
-----[1]
-----[a]
---<Print>
----[a]
-<BLOCK>
--<VariableDeclaration>
```

```
---[boolean]
---[b]
--<Assign>
---[b]
---[true]
--<While>
---<isNotEqual>
----[b]
----[false]
---<BLOCK>
----<Assign>
-----[b]
-----[false]
----<VariableDeclaration>
-----[string]
-----[a]
----<Assign>
-----[a]
-----["a"]
----<Print>
-----[a]
-<While>
--<isNotEqual>
---[a]
---[3]
--<BLOCK>
---<Print>
----[a]
---<Assign>
----[a]
----<Addition>
-----[1]
-----[a]
-<Print>
--["bye"]

Program 1 Symbol Table
---------------------------
Name  Type     Scope  Line
---------------------------
a     int      0      2
b     boolean  2      11
a     string   3      16


CODE GENERATION: Beginning Code Generation on Program 1 ...
CODE GENERATION: Storing value: false in heap at location: 250
CODE GENERATION: Storing value: true in heap at location: 245
CODE GENERATION: Adding Variable Declaration of Variable: a
CODE GENERATION: Assigning Variable a to value: 1
CODE GENERATION: Comparing values: a and 1 in inequality operation.
CODE GENERATION: Storing Addition Operation: 1 + a in variable: a
CODE GENERATION: Printing variable: a
CODE GENERATION: Adding Variable Declaration of Variable: b
CODE GENERATION: Assigning Variable b to value: true
CODE GENERATION: Comparing values: b and false in inequality operation.
```

```
 CODE GENERATION: Assigning Variable b to value: false
 CODE GENERATION: Adding Variable Declaration of Variable: a
 CODE GENERATION: Storing value: a in heap at location: 243
 CODE GENERATION: Assigning Variable a to value: "a"
 CODE GENERATION: Printing variable: a
 CODE GENERATION: Comparing values: a and 3 in inequality operation.
 CODE GENERATION: Printing variable: a
 CODE GENERATION: Storing Addition Operation: 1 + a in variable: a
 CODE GENERATION: Storing value: bye in heap at location: 239
 CODE GENERATION: Printing value: "bye"
 CODE GENERATION: Adding Break Statement
 CODE GENERATION: Backpatching Static Variable Placeholder T0XX With Memory Address
E2
 CODE GENERATION: Backpatching Static Variable Placeholder T1XX With Memory Address
E3
 CODE GENERATION: Backpatching Static Variable Placeholder T2XX With Memory Address
E4
 CODE GENERATION: Backpatching Static Variable Placeholder T3XX With Memory Address
E5
 CODE GENERATION: Backpatching Static Variable Placeholder T4XX With Memory Address
E6
 CODE GENERATION: Backpatching Static Variable Placeholder T5XX With Memory Address
E7
 CODE GENERATION: Backpatching Static Variable Placeholder T6XX With Memory Address
E8
 CODE GENERATION: Backpatching Static Variable Placeholder T7XX With Memory Address
E9
 CODE GENERATION: Backpatching Static Variable Placeholder T8XX With Memory Address
EA
 CODE GENERATION: Backpatching Static Variable Placeholder T9XX With Memory Address
EB
 CODE GENERATION: Backpatching Jump Variable Placeholder J0 Forward 2D Addresses
 CODE GENERATION: Backpatching Jump Variable Placeholder J1 Forward BB Addresses
 CODE GENERATION: Backpatching Jump Variable Placeholder J2 Forward 26 Addresses
 CODE GENERATION: Backpatching Jump Variable Placeholder J3 Forward C2 Addresses
 CODE GENERATION: Backpatching Jump Variable Placeholder J4 Forward 2D Addresses
 CODE GENERATION: Backpatching Jump Variable Placeholder J5 Forward BB Addresses
 Program 1 Code Generation Passed With 0 error(s)

 Program 1 Static Variable Table
 --------------------------------
 Name  Temp   Address  Scope
 --------------------------------
 a     T0XX   E2        0
 0     T1XX   E3       -1
 1     T2XX   E4       -1
 2     T3XX   E5       -1
 b     T4XX   E6        2
 3     T5XX   E7       -1
 a     T6XX   E8        3
 4     T7XX   E9       -1
 5     T8XX   EA       -1
 6     T9XX   EB       -1

 Program 1 Jump Table
 ---------------------
```

```
Temp  Distance
----------------------
J0    2D
J1    BB
J2    26
J3    C2
J4    2D
J5    BB

Program 1 Machine Code:
A9 00 8D E2 00 A9 01 8D E2 00 A2 01 EC E2 00 A9
F5 8D E3 00 D0 05 A9 FA 8D E3 00 A2 F5 EC E3 00
D0 2D A9 F5 8D E3 00 A9 01 8D E4 00 A9 00 6D E2
00 6D E4 00 8D E5 00 AD E5 00 8D E2 00 AC E2 00
A2 01 FF A9 00 8D 00 00 A2 01 EC 00 00 D0 BB A9
FA 8D E6 00 A9 F5 8D E6 00 A2 FA EC E6 00 A9 F5
8D E7 00 D0 05 A9 FA 8D E7 00 A2 F5 EC E7 00 D0
26 A9 F5 8D E7 00 A9 FA 8D E6 00 A9 00 8D E8 00
A9 F3 8D E8 00 AC E8 00 A2 02 FF A9 00 8D 00 00
A2 01 EC 00 00 D0 C2 A2 03 EC E2 00 A9 F5 8D E9
00 D0 05 A9 FA 8D E9 00 A2 F5 EC E9 00 D0 2D A9
F5 8D E9 00 AC E2 00 A2 01 FF A9 01 8D EA 00 A9
00 6D E2 00 6D EA 00 8D EB 00 AD EB 00 8D E2 00
A9 00 8D 00 00 A2 01 EC 00 00 D0 BB A0 EF A2 02
FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 62
79 65 00 61 00 74 72 75 65 00 66 61 6C 73 65 00
```

This program consists of three separate while loops. First in scope 0, int i is declared and set to 1. Then, the first while is when a != 1, since a = 1 it skips over that loop. Then it enters scope 2 where boolean b is declared and set to true. The conditional for the next while is b != false. Since b = true, it enters the loop (scope 3). The variable b is set to false and a string a is declared and set to "a" and printed. We then loop back to the top of the loop and reach the conditional b != false. Because b is now equal to false it skips over the loop. We reach another conditional (back to scope 2) that checks if a != 3. There is no variable a in scope 2, but there is in the parent scope 0 where a = 1. Since a != 3 we enter the loop where a is printed and incremented by one (now equal to 2). Then we loop back to the top of the loop and check again if a != 3. Since it is equal to 2, we enter the loop again and print a and increment it by one. Then we jump back to the top of the loop again and check if a != 3. Since a = 3, we jump past the loop and finally output "bye". The final output is: a12bye.