

**LAB # 11**

**Submitted By:**

Ramsha Kokab

FA21-BCS-010

**Course Instructor:**

Mr. Syed Bilal Haider Bukhari

**Course:**

CC- [Lab]

**Date:**

13th Dec, 2024

**DEPARTMENT OF COMPUTER SCIENCE**

**COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS**

**Q: Implement Syntax Directed Translation for each node of the above grammar.**

using System;

using System.Collections.Generic;

namespace Lab11

{

// Token class to store type, value, and position

public class Token

{

public string Type { get; }

public string Value { get; }

public int Line { get; }

public int Column { get; }

public Token(string type, string value, int line, int column)

{

Type = type;

Value = value;

Line = line;

Column = column;

}

public override string ToString()

{

return $"{Type} ({Line},{Column}): {Value}";

}

}

// Lexer to generate tokens

public class Lexer

{

private readonly string[] inputLines;

private int currentLine;

private int currentColumn;

public Lexer(string[] input)

{

inputLines = input;

currentLine = 0;

currentColumn = 0;

}

// Simulates token generation from input

public List<Token> GenerateTokens()

{

var tokens = new List<Token>

{

new Token("INT", null, 1, 1),

new Token("MAIN", null, 1, 5),

new Token("LPAREN", null, 1, 9),

new Token("RPAREN", null, 1, 10),

new Token("LBRACE", null, 1, 11),

new Token("INT", null, 2, 3),

new Token("ID", "x", 2, 7),

new Token("SEMI", null, 2, 8),

new Token("ID", "x", 3, 3),

new Token("SEMI", null, 3, 4),

new Token("ID", "x", 4, 3),

new Token("ASSIGN", null, 4, 5),

new Token("INT\_CONST", "2", 4, 7),

new Token("PLUS", null, 4, 9),

new Token("INT\_CONST", "5", 4, 11),

new Token("PLUS", null, 4, 13),

new Token("LPAREN", null, 4, 15),

new Token("INT\_CONST", "4", 4, 16),

new Token("TIMES", null, 4, 18),

new Token("INT\_CONST", "8", 4, 20),

new Token("RPAREN", null, 4, 21),

new Token("PLUS", null, 4, 23),

new Token("CHAR\_CONST", "l", 4, 27),

new Token("DIV", null, 4, 29),

new Token("FLOAT\_CONST", "9.0", 4, 31),

new Token("SEMI", null, 4, 34),

new Token("IF", null, 5, 3),

new Token("LPAREN", null, 5, 5),

new Token("ID", "x", 5, 6),

new Token("PLUS", null, 5, 8),

new Token("ID", "y", 5, 10),

new Token("RPAREN", null, 5, 11),

new Token("LBRACE", null, 5, 12),

new Token("IF", null, 6, 5),

new Token("LPAREN", null, 6, 8),

new Token("ID", "x", 6, 10),

new Token("NEQ", null, 6, 11),

new Token("INT\_CONST", "4", 6, 14),

new Token("RPAREN", null, 6, 15),

new Token("LBRACE", null, 6, 16),

new Token("ID", "x", 7, 7),

new Token("ASSIGN", null, 7, 9),

new Token("INT\_CONST", "6", 7, 11),

new Token("SEMI", null, 7, 12),

new Token("ID", "y", 8, 7),

new Token("ASSIGN", null, 8, 9),

new Token("INT\_CONST", "10", 8, 11),

new Token("SEMI", null, 8, 13),

new Token("ID", "i", 9, 7),

new Token("ASSIGN", null, 9, 9),

new Token("INT\_CONST", "11", 9, 11),

new Token("SEMI", null, 9, 13),

new Token("RBRACE", null, 10, 5),

new Token("RBRACE", null, 11, 3),

new Token("RBRACE", null, 12, 1),

new Token("EOF", null, 12, 2)

};

return tokens;

}

}

// Main Program

class Program

{

static void Main(string[] args)

{

var inputLines = new[]

{

// Simulated input for testing

"INT MAIN() {",

" INT x;",

" x;",

" x = 2 + 5 + (4 \* 8) + 'l' / 9.0;",

" IF (x + y) {",

" IF (x != 4) {",

" x = 6;",

" y = 10;",

" i = 11;",

" }",

" }",

"}"

};

Lexer lexer = new Lexer(inputLines);

var tokens = lexer.GenerateTokens();

Console.WriteLine("Generated Tokens:");

foreach (var token in tokens)

{

Console.WriteLine(token);

}

Console.WriteLine("\nPress any key to exit...");

Console.ReadKey();

}

}

}

**Output:**

