Question2:

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

namespace MiniLanguageLexerConsole

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Mini-Language Lexical Analyzer (Console)");

Console.WriteLine("Enter code (press Enter twice to analyze):");

// Read multi-line input until double Enter

string inputCode = ReadMultiLineInput();

// Analyze and display results

List<VariableToken> tokens = LexicalAnalyzer.AnalyzeCode(inputCode);

DisplayResults(tokens);

}

static string ReadMultiLineInput()

{

string input = "";

string line;

while (!string.IsNullOrWhiteSpace(line = Console.ReadLine()))

{

input += line + Environment.NewLine;

}

return input;

}

static void DisplayResults(List<VariableToken> tokens)

{

Console.WriteLine("\nResults:");

Console.WriteLine("-------------------------------");

Console.WriteLine("| {0,-10} | {1,-12} | {2,-8} |",

"VarName", "SpecialSymbol", "Type");

Console.WriteLine("-------------------------------");

foreach (var token in tokens)

{

Console.WriteLine("| {0,-10} | {1,-12} | {2,-8} |",

token.VarName, token.SpecialSymbol, token.TokenType);

}

Console.WriteLine("-------------------------------");

}

}

public static class LexicalAnalyzer

{

public static List<VariableToken> AnalyzeCode(string code)

{

List<VariableToken> tokens = new List<VariableToken>();

string pattern = @"\b(var\s+|float\s+)?([abc][a-zA-Z]\*\d+)\s\*=\s\*([^;]+)[;$]";

foreach (Match match in Regex.Matches(code, pattern))

{

string varName = match.Groups[2].Value;

string value = match.Groups[3].Value.Trim();

string specialSymbols = Regex.Replace(value, @"[\w\.]", "");

if (!string.IsNullOrEmpty(specialSymbols))

{

tokens.Add(new VariableToken

{

VarName = varName,

SpecialSymbol = specialSymbols,

TokenType = match.Groups[1].Value.Contains("float") ? "Float" : "Integer"

});

}

}

return tokens;

}

}

public class VariableToken

{

public string VarName { get; set; }

public string SpecialSymbol { get; set; }

public string TokenType { get; set; }

}

}

**Output:**

