**LAB (4) TASK (1)**

**Code:**

using System;

using System.Collections.Generic;

class LexicalAnalyzer

{

static string[] buffer = new string[2]; // Two buffers

static int currentBuffer = 0;

static int pointer = 0;

static string sourceCode = "int main() { int a = 5 + 10; return a; }";

static HashSet<string> keywords = new HashSet<string> { "int", "return", "main" };

static HashSet<char> operators = new HashSet<char> { '+', '-', '\*', '/', '=', '<', '>' };

static HashSet<char> punctuation = new HashSet<char> { ';', ',', '(', ')', '{', '}' };

static void Main()

{

Console.WriteLine("Source Code:");

Console.WriteLine(sourceCode);

Console.WriteLine("\nTokens:");

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

LoadBuffers();

string token = "";

char ch;

while ((ch = NextChar()) != '\0')

{

if (char.IsWhiteSpace(ch))

{

ProcessToken(token);

token = "";

}

else if (operators.Contains(ch))

{

ProcessToken(token);

token = "";

Console.WriteLine("Operator\t: " + ch);

}

else if (punctuation.Contains(ch))

{

ProcessToken(token);

token = "";

Console.WriteLine("Punctuation\t: " + ch);

}

else

{

token += ch; } }

ProcessToken(token);

}

static void LoadBuffers()

{

int mid = sourceCode.Length / 2;

buffer[0] = sourceCode.Substring(0, mid);

buffer[1] = sourceCode.Substring(mid);

}

static char NextChar()

{

if (pointer >= buffer[currentBuffer].Length)

{

currentBuffer++;

pointer = 0;

if (currentBuffer > 1)

return '\0'; // End of source

return NextChar();

}

return buffer[currentBuffer][pointer++];

}

static void ProcessToken(string token)

{

if (string.IsNullOrWhiteSpace(token))

return;

if (keywords.Contains(token))

Console.WriteLine("Keyword\t\t: " + token);

else if (char.IsDigit(token[0]))

Console.WriteLine("Number\t\t: " + token);

else

Console.WriteLine("Identifier\t: " + token); } }

**Output:**

