**LAB (11) TASK (1)**

**Code:**

using System;

using System.Collections.Generic;

class Program

{

static Queue<string> tokens = new Queue<string>();

static Queue<string> lexemes = new Queue<string>();

static string variable = "", value = "";

static void Main()

{

Console.Write("Enter input (e.g., int x = 10;): ");

string input = Console.ReadLine();

// 🔧 Ensure semicolon at end if missing

if (!input.Trim().EndsWith(";"))

input = input.Trim() + ";";

Tokenize(input);

bool isValid = S();

if (isValid && tokens.Count == 0)

{

Console.WriteLine("\n✅ Input is valid.");

Console.WriteLine("📘 Translation:");

if (!string.IsNullOrEmpty(variable))

Console.WriteLine($"Declare/Use variable: {variable}");

if (!string.IsNullOrEmpty(value))

Console.WriteLine($"Assign value: {value} to variable {variable}");

}

else

{

Console.WriteLine("\n❌ Input is NOT valid.");

}

}

static void Tokenize(string input)

{

// Replace symbols with spaced versions so they tokenize correctly

input = input.Replace("=", " = ").Replace(";", " ; ");

string[] parts = input.Split(new[] { ' ', '\t' }, StringSplitOptions.RemoveEmptyEntries);

foreach (var part in parts)

{

lexemes.Enqueue(part);

if (part == "int")

tokens.Enqueue("int");

else if (part == "=")

tokens.Enqueue("=");

else if (part == ";")

tokens.Enqueue(";");

else if (int.TryParse(part, out \_))

tokens.Enqueue("num");

else

tokens.Enqueue("id");

}

}

static bool S()

{

return A() && B();

}

static bool A()

{

if (tokens.Count == 0) return false;

string current = tokens.Peek();

if (current == "int")

{

tokens.Dequeue(); lexemes.Dequeue(); // consume 'int'

if (tokens.Count > 0 && tokens.Peek() == "id")

{

tokens.Dequeue();

variable = lexemes.Dequeue(); // get variable name

return true;

}

return false;

}

else if (current == "id")

{

tokens.Dequeue();

variable = lexemes.Dequeue(); // use existing variable

return true;

}

return false;

}

static bool B()

{

if (tokens.Count < 2) return false;

if (tokens.Peek() == "=")

{

tokens.Dequeue(); lexemes.Dequeue(); // consume '='

if (C())

{

if (tokens.Count > 0 && tokens.Peek() == ";")

{

tokens.Dequeue(); lexemes.Dequeue(); // consume ';'

return true;

}

}

}

return false;

}

static bool C()

{

if (tokens.Count == 0) return false;

string current = tokens.Peek();

if (current == "num" || current == "id")

{

tokens.Dequeue();

value = lexemes.Dequeue(); // get value

return true;

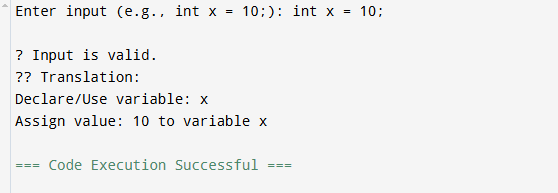
}

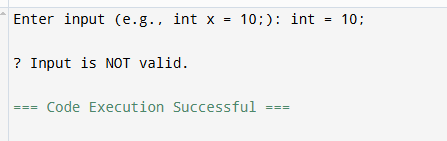
return false;

}

}

**Output:**

****

****