

**LAB # 10**

**Submitted By:**

Ramsha Kokab

FA21-BCS-010

**Course Instructor:**

Mr. Syed Bilal Haider Bukhari

**Course:**

CC- [Lab]

**Date:**

29th Nov, 2024

**DEPARTMENT OF COMPUTER SCIENCE**

**COMSATS UNIVERSITY ISLAMABAD, ATTOCK CAMPUS**

**Q: Understand the SLR code, mentioned in lab activity 1, written with the help of dictionary and stack classes of C#, execute the same with the output.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab10\_SLR

{

internal class Program

{

static void Main(string[] args)

{// Define the parsing table

var parseTable = new Dictionary<string, Dictionary<string, string>>

{

{ "0", new Dictionary<string, string> { { "begin", "S2" }, { "Program", "1" } } },

{ "2", new Dictionary<string, string> { { "int", "S3" } } },

{ "3", new Dictionary<string, string> { { "a", "S4" }, { "Var", "5" } } },

{ "4", new Dictionary<string, string> { { "=", "S6" } } },

{ "6", new Dictionary<string, string> { { "5", "S7" }, { "Const", "8" } } },

{ "7", new Dictionary<string, string> { { ";", "R3" } } },

{ "8", new Dictionary<string, string> { { ";", "R1" } } },

{ "5", new Dictionary<string, string> { { ";", "R2" } } },

{ "1", new Dictionary<string, string> { { "$", "Accept" } } }

};

// Define the rules

var rules = new Dictionary<int, (string lhs, int rhsCount)>

{

{ 1, ("Program", 6) },

{ 2, ("DecS", 4) },

{ 3, ("AssS", 5) }

};

// Initialize the stack and input

var stack = new Stack<string>();

stack.Push("0");

string input = "begin int a = 5 ; $";

var tokens = input.Split(' ');

int pointer = 0;

Console.WriteLine("Parsing started...\n");

Console.WriteLine($"Input: {input}");

while (true)

{

string currentState = stack.Peek();

string currentToken = tokens[pointer];

// Check if the action exists in the table

if (!parseTable.ContainsKey(currentState) || !parseTable[currentState].ContainsKey(currentToken))

{

Console.WriteLine($"Error: Unable to parse. Missing transition for state {currentState} and token {currentToken}.");

break;

}

string action = parseTable[currentState][currentToken];

if (action.StartsWith("S"))

{

// Shift: Push token and state

stack.Push(currentToken);

stack.Push(action.Substring(1));

pointer++;

}

else if (action.StartsWith("R"))

{

// Reduce: Apply the rule

int ruleIndex = int.Parse(action.Substring(1));

var (lhs, rhsCount) = rules[ruleIndex];

// Pop 2 \* rhsCount for both tokens and states

for (int i = 0; i < rhsCount \* 2; i++) stack.Pop();

string newState = stack.Peek();

// Debugging: Check if Goto entry exists

if (!parseTable.ContainsKey(newState) || !parseTable[newState].ContainsKey(lhs))

{

Console.WriteLine($"Error: Missing Goto entry for state {newState} and symbol {lhs}.");

break;

}

// Push LHS and goto state

stack.Push(lhs);

stack.Push(parseTable[newState][lhs]);

}

else if (action == "Accept")

{

Console.WriteLine("Parsing successful!");

break;

}

Console.WriteLine($"Stack: {string.Join(" ", stack)}");

}

// Pause for user to exit

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

Console.ReadKey();

}}}

**Output:**

