

**Lab # 7**

**Submitted By:** Sana Naz (SP21-BCS-031)

**Course Instructor:** Mr. Bilal bukhari

**Course:** Compiler constrcution

**Date:** 18th Octuber, 2024

**Lab Task :** Write a code for any given grammar that satisfy the criterion of JAVA language constructs.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab7

{

class Program

{

static int limit, x = 0;

static char[,] production = new char[10, 10];

static char[] array = new char[10];

static void Main(string[] args)

{

// Initialize production array with default values ('-')

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

production[i, j] = '-';

}

}

// Input the number of productions from the user

Console.WriteLine("\nEnter Total Number of Productions:\t");

limit = Convert.ToInt32(Console.ReadLine());

// Input each production rule from the user

for (int count = 0; count < limit; count++)

{

Console.WriteLine("\nValue of Production Number {0}:\t", count + 1);

string temp = Console.ReadLine();

for (int i = 0; i < temp.Length; i++)

{

production[count, i] = temp[i];

}

}

// Repeatedly ask the user for non-terminal to find Follow set

char option, ch;

do

{

x = 0;

Console.WriteLine("\nEnter Production Value to Find Follow:\t");

ch = Console.ReadKey().KeyChar;

Console.WriteLine();

find\_follow(ch);

Console.WriteLine("\nFollow Value of {0}:\t{{", ch);

for (int count = 0; count < x; count++)

{

Console.Write(array[count]);

}

Console.WriteLine("}");

Console.Write("To Continue, Press Y:\t");

option = Console.ReadKey().KeyChar;

Console.WriteLine();

} while (option == 'y' || option == 'Y');

// Print all productions (for debugging purposes)

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 10; j++)

{

Console.Write(production[i, j]);

}

Console.WriteLine();

}

Console.ReadKey();

}

static void find\_follow(char ch)

{

for (int i = 0; i < limit; i++)

{

int length = production[i, 0].ToString().Length;

// If the current non-terminal is the start symbol, add '$' to Follow set

if (Convert.ToChar(production[0, 0]).Equals(ch))

{

array\_manipulation('$');

}

// Loop through the production to find occurrences of the given non-terminal

for (int j = 2; j < length; j++)

{

if (Convert.ToChar(production[i, j]).Equals(ch))

{

// If the next symbol exists, find its First set

if (j + 1 < length && !production[i, j + 1].Equals('\0'))

{

find\_first(Convert.ToChar(production[i, j + 1]));

}

// If the symbol is at the end of the production, find Follow of the LHS non-terminal

if (j + 1 == length && !production[i, 0].Equals(ch))

{

find\_follow(Convert.ToChar(production[i, 0]));

}

}

}

}

}

static void find\_first(char ch)

{

// Check if the symbol is uppercase (non-terminal)

if (char.IsUpper(ch))

{

array\_manipulation(ch);

return;

}

// Loop through productions to find First set for the non-terminal

for (int k = 0; k < limit; k++)

{

if (production[k, 0].Equals(ch))

{

if (production[k, 2].Equals('$'))

{

find\_follow(Convert.ToChar(production[k, 0]));

}

else if (char.IsLower(production[k, 2]))

{

array\_manipulation(production[k, 2]);

}

else

{

find\_first(production[k, 2]);}

}}}

static void array\_manipulation(char ch)

{

// Add the character to the array only if it doesn't already exist

for (int count = 0; count < x; count++)

{

if (array[count] == ch)

return; }

array[x++] = ch;

}

}

}

**Output:**

