Question4:

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main()

{

// Step 1: Take input from user

Console.Write("Enter number of non-terminals: ");

int n = int.Parse(Console.ReadLine());

var grammar = new Dictionary<string, List<List<string>>>();

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

{

Console.Write($"Enter production for non-terminal {i + 1}: ");

string input = Console.ReadLine();

var parts = input.Split(new[] { "->" }, StringSplitOptions.RemoveEmptyEntries);

string lhs = parts[0].Trim();

string[] rhsOptions = parts[1].Split('|');

var productions = new List<List<string>>();

foreach (string option in rhsOptions)

{

List<string> symbols = option.Trim().Split(' ').ToList();

productions.Add(symbols);

}

grammar[lhs] = productions;

}

var firstSets = ComputeFirstSets(grammar);

var followSets = ComputeFollowSets(grammar, firstSets);

Console.WriteLine("\nFIRST Sets:");

foreach (var kvp in firstSets)

{

Console.WriteLine($"FIRST({kvp.Key}) = {{ {string.Join(", ", kvp.Value)} }}");

}

Console.WriteLine("\nFOLLOW Sets:");

foreach (var kvp in followSets)

{

Console.WriteLine($"FOLLOW({kvp.Key}) = {{ {string.Join(", ", kvp.Value)} }}");

}

}

static Dictionary<string, HashSet<string>> ComputeFirstSets(Dictionary<string, List<List<string>>> grammar)

{

var first = new Dictionary<string, HashSet<string>>();

foreach (var nt in grammar.Keys)

first[nt] = new HashSet<string>();

bool changed;

do

{

changed = false;

foreach (var nt in grammar.Keys)

{

foreach (var production in grammar[nt])

{

for (int i = 0; i < production.Count; i++)

{

string symbol = production[i];

if (!grammar.ContainsKey(symbol)) // Terminal

{

if (first[nt].Add(symbol))

changed = true;

break;

}

foreach (var f in first[symbol])

{

if (f != "ε" && first[nt].Add(f))

changed = true;

}

if (!first[symbol].Contains("ε"))

break;

if (i == production.Count - 1 && first[nt].Add("ε"))

changed = true;

}

}

}

} while (changed);

return first;

}

static Dictionary<string, HashSet<string>> ComputeFollowSets(Dictionary<string, List<List<string>>> grammar, Dictionary<string, HashSet<string>> firstSets)

{

var follow = new Dictionary<string, HashSet<string>>();

foreach (var nt in grammar.Keys)

follow[nt] = new HashSet<string>();

string startSymbol = grammar.Keys.First();

follow[startSymbol].Add("$");

bool changed;

do

{

changed = false;

foreach (var lhs in grammar.Keys)

{

foreach (var production in grammar[lhs])

{

for (int i = 0; i < production.Count; i++)

{

string symbol = production[i];

if (!grammar.ContainsKey(symbol))

continue;

bool epsilonInAll = true;

for (int j = i + 1; j < production.Count; j++)

{

string next = production[j];

if (!grammar.ContainsKey(next))

{

if (follow[symbol].Add(next))

changed = true;

epsilonInAll = false;

break;

}

foreach (var f in firstSets[next])

{

if (f != "ε" && follow[symbol].Add(f))

changed = true;

}

if (!firstSets[next].Contains("ε"))

{

epsilonInAll = false;

break;

}

}

if (epsilonInAll)

{

foreach (var f in follow[lhs])

{

if (follow[symbol].Add(f))

changed = true;

}

}

}

}

}

} while (changed);

return follow;

}

}

Output:

