**COMSATS UNIVERSITY ISLAMABAD**

**(ATTOCK CAMPUS)**

****

**Lab Mid**

**Subject:**

**CC Lab**

**Submitted to:**

**Sir Bilal Bukhari**

**Submitted by:**

Kashif Hussain

Fa20-bcs-019

**Dated**: 26th Oct,2023

### Question :1

### Describe functioning of regex C# library , give examples of patterns,seperators and anchors e.t.c.

### Answer:

# **Functioning of Regex in C# Library:**

# Regular expressions in C# are implemented through the “System.Text.RegularExpressions” namespace.

# The **Regex class** provides methods for working with regular expressions, allowing you to search, match, and manipulate strings based on specific patterns.

# **Examples of Regex Elements:**

# **Patterns:**

# Patterns are sequences of characters that define a search pattern.

# **For example**:

# \d{3}-\d{2}-\d{4} is a pattern that matches a social security number.

# **Separators**:

# Separators can be any characters used to divide or distinguish parts of a string.

# **For instance:**

# in the pattern cat|dog, the pipe symbol (|) acts as a separator, allowing either 'cat' or 'dog' to match.

# **Anchors**:

# Anchors specify positions in the input string. ^ is an anchor that matches the start of a string, and \b represents a word boundary.

# **Quantifiers**:

# Quantifiers define how many instances of a character or group are expected.

# **For example**: {2,4} specifies 2 to 4 occurrences.

# **Escape Sequences**: Escape sequences allow matching of specific characters or predefined character classes. \d matches any digit, and \s matches any whitespace character.

# **Example Patterns**:

# **Pattern**: \d{3}-\d{2}-\d{4}

# **Description:** Matches social security numbers in the format ###-##-####.

# **Pattern**: \b[A-Z][a-z]+\b

# **Description**: Matches capitalized words.

# **Pattern:** ^\d{4}$

# **Description**: Matches exactly 4 digits at the start of the string.

# **Example Separators:**

# **Separator**: |

# Description: Separates alternative patterns.

# **For instance**, cat|dog matches either 'cat' or 'dog'.

# **Example Anchors:**

# **Anchor**: ^

# **Description**: Matches the start of a string.

# **For example**: ^Hello matches 'Hello' only at the beginning of the input string.

### Question :2

For the given grammar create LL(1) or recursive descent parser

List -> Item Rest

Rest -> , Item Rest | ε

Item -> id | num | string

### Answer:

class Parser

{

private List<string> tokens;

private int index;

public Parser(List<string> input)

{

tokens = input;

index = 0;

}

// Start the parsing process

public void Parse()

{

List();

if (index == tokens.Count)

Console.WriteLine("Parsing successful!");

else

Console.WriteLine("Parsing failed: Unexpected input.");

}

private void List()

{

Item();

Rest();

}

private void Rest()

{

if (index < tokens.Count && tokens[index] == ",")

{

index++;

Item();

Rest();

}

// epsilon (empty) case

}

private void Item()

{

if (index < tokens.Count && (tokens[index] == "id" || tokens[index] == "num" || tokens[index] == "string"))

{

index++;

}

else

{

Console.WriteLine("Parsing failed: Expected 'id', 'num', or 'string'.");

Environment.Exit(1);

}

}

}

class Program

{

static void Main()

{

List<string> inputTokens = new List<string> { "id", ",", "num", ",", "string" };

Parser parser = new Parser(inputTokens);

parser.Parse();

}

### Output:

