**CC LAB MID**

****

**Submitted By:**

Sharafat Noor  
(SP21-BCS-036)

**Submitted To:**

Sir Bilal Bukhari

**Subject:**

Compiler Construction

**Date:**

7th April 2024

**Question No. 1:**

In C#, the regex library provides a powerful toolset for working with regular expressions. It's primarily accessed through the System.Text.RegularExpressions namespace. Here's a brief overview of its features:

**Regex Class:** The Regex class is the main entry point for working with regular expressions in C#. It provides methods for pattern matching, replacing, and splitting strings based on regex patterns.

**Pattern Syntax:** C# regex supports the standard regex syntax along with some additional features specific to .NET. This includes character classes, quantifiers, anchors, grouping constructs, alternation, and more.

**Match Object:** When a regex pattern matches a string, it returns a Match object containing information about the match. It represents a successful match.

**Matches Collection:** it represents Collection of all matches in a string.

**Replacement:** The Regex.Replace method allows you to perform regex-based string replacements. You can specify a replacement pattern that may include references to captured groups.

**Options:** The Regex Options enum allows you to specify various options for regex matching, such as case sensitivity, multiline mode, and whether to ignore whitespace in the pattern.

**Validation:** Regex can also be used for string validation tasks such as checking if a string matches a certain pattern or contains certain characters.

Overall, the regex library in C# provides a comprehensive set of tools for working with regular expressions, making it easier to perform complex string manipulation tasks.

**Question No.2**

**CODE:**

#include <iostream>

#include <string>

using namespace std;

class Parser {

private:

string input\_string;

size\_t index;

char current\_token;

public:

Parser(const string& input) : input\_string(input), index(0), current\_token(input[0]) {}

bool match(char expected\_token) {

if (current\_token == expected\_token) {

index++;

if (index < input\_string.length()) {

current\_token = input\_string[index];

}

return true;

}

return false;

}

bool parse\_S() {

if (parse\_X() && match('$')) {

return true;

}

return false;

}

bool parse\_X() {

if (parse\_Y() && parse\_X\_prime()) {

return true;

}

return false;

}

bool parse\_X\_prime() {

if (match('%')) {

if (parse\_Y() && parse\_X\_prime()) {

return true;

}

}

return true;

}

bool parse\_Y() {

if (parse\_Z() && parse\_Y\_prime()) {

return true;

}

return false;

}

bool parse\_Y\_prime() {

if (match('&')) {

if (parse\_Z() && parse\_Y\_prime()) {

return true;

}

}

return true;

}

bool parse\_Z() {

if (match('k') && parse\_X() && match('k')) {

return true;

}

else if (match('g')) {

return true;

}

return false;

}

bool parse() {

return parse\_S();

}

};

int main() {

string input;

cout << "Enter a string to parse: ";

cin >> input;

Parser parser(input);

if (parser.parse()) {

cout << "String is in the language" << endl;

}

else {

cout << "String is not in the language" << endl;

}

return 0;

}

**Question No.3**

**CODE:**

using System;

using System.Text;

using System.Linq;

using System.Collections.Generic;

public class PasswordGenerator

{

private static Random random = new Random();

private const string UppercaseLetters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

private const string Numbers = "0123456789";

private const string SpecialCharacters = "!@#$%^&\*";

public static string GeneratePassword(string firstName, string lastName, string registrationNumber)

{

if (string.IsNullOrEmpty(firstName) || string.IsNullOrEmpty(lastName) || registrationNumber.Length < 2)

{

throw new ArgumentException("Invalid input parameters for generating password.");

}

// At least one uppercase alphabet - using the first letter of the first name

string password = firstName.Substring(0, 1).ToUpper();

// At least 4 numbers, two of which are from the registration number

string numbers = registrationNumber.Substring(0, 2) // Taking first two numbers from registration

+ RandomString(Numbers, 2); // Adding two more random numbers

// At least 2 special characters

string specialChars = RandomString(SpecialCharacters, 2);

// Initials of first and last name

string initials = firstName.Substring(0, 1) + lastName.Substring(0, 1);

// All odd letters of your first name

string oddLettersFirstName = new string(firstName.Where((c, i) => i % 2 == 0).ToArray());

// All even letters of your last name

string evenLettersLastName = new string(lastName.Where((c, i) => i % 2 != 0).ToArray());

// Combine all elements

string combined = password + numbers + specialChars + initials + oddLettersFirstName + evenLettersLastName;

// If combined string is longer than 16 characters, trim it; otherwise, fill with random letters up to 16

if (combined.Length > 16)

{

return combined.Substring(0, 16);

}

else

{

return combined + RandomString(UppercaseLetters, 16 - combined.Length);

}

}

private static string RandomString(string chars, int length)

{

return new string(Enumerable.Repeat(chars, length)

.Select(s => s[random.Next(s.Length)]).ToArray());

}

static void Main(string[] args)

{

Console.Write("Enter your first name: ");

string firstName = Console.ReadLine();

Console.Write("Enter your last name: ");

string lastName = Console.ReadLine();

Console.Write("Enter your registration number (at least two digits): ");

string registrationNumber = Console.ReadLine();

// Basic validation

if (string.IsNullOrWhiteSpace(firstName) || string.IsNullOrWhiteSpace(lastName) || registrationNumber.Length < 2)

{

Console.WriteLine("Invalid input. Please ensure you've entered the correct details.");

return;

}

try {

string password = GeneratePassword(firstName, lastName, registrationNumber);

Console.WriteLine($"Generated Password: {password}");

}

catch (ArgumentException ex)

{

Console.WriteLine($"Error: {ex.Message}"); } }}

**SCREENSHOT ATTACHED BELOW:**

