**Question no 1:-**

he regex library in C# is part of the .NET framework and provides functionality for working with regular expressions. Here's a brief overview:

Namespace: The regex library is located in the System.Text.RegularExpressions namespace.

Classes: The main class used for working with regular expressions is Regex. This class provides methods for pattern matching, replacing, and splitting strings based on regular expressions.

Pattern Syntax: C# regex patterns follow the standard regular expression syntax, supporting a wide range of features including character classes, quantifiers, anchors, capturing groups, and more.

Matches: The Match class represents a single match in a string. It contains information about the matched text and any captured groups.

Methods: The Regex class provides methods like Match, Matches, Replace, Split, etc., for performing various operations on strings using regular expressions.

Options: Regex options can be specified using RegexOptions enumeration to modify the behavior of the regular expression engine, such as case insensitivity, single-line mode, etc.

**Question no 2:-**

#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;**

**}**

**Quesstion no 3:-**

using System.Linq;

using System.Text;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Welcome to the Password Generator!");

Console.WriteLine("Please enter your first name:");

string firstName = Console.ReadLine();

Console.WriteLine("Please enter your last name:");

string lastName = Console.ReadLine();

Console.WriteLine("Please enter your registration numbers:");

string registrationNumbers = Console.ReadLine();

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

Console.WriteLine("Generated Password: " + password);

}

static string GeneratePassword(string firstName, string lastName, string registrationNumbers)

{

StringBuilder password = new StringBuilder();

password.Append(char.ToUpper(firstName[0]));

password.Append(char.ToUpper(lastName[0]));

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

{

if (i % 2 == 0)

password.Append(firstName[i]);

}

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

{

if (i % 2 != 0)

password.Append(lastName[i]);

}

password.Append((char)('A' + new Random().Next(0, 26)));

var selectedNumbers = registrationNumbers.Where(char.IsDigit).OrderBy(n => Guid.NewGuid()).Take(2);

foreach (var number in selectedNumbers)

{

password.Append(number);

}

for (int i = 0; i < 2; i++)

{

password.Append(new Random().Next(0, 10));

}

string specialChars = "!@#$%^&\*()\_+-=[]{}|;:,.<>?";

for (int i = 0; i < 2; i++)

{

password.Append(specialChars[new Random().Next(0, specialChars.Length)]);

}

string shuffledPassword = new string(password.ToString().OrderBy(x => Guid.NewGuid()).ToArray());

if (shuffledPassword.Length > 16)

{

shuffledPassword = shuffledPassword.Substring(0, 16);

}

return shuffledPassword;

}

}