**Lab Mid**



**Submitted By**

Hammas Rashid

**Roll No**

SP21-BCS-011

**Submitted To**

Sir Bilal Bukhari

**Subject**

Compiler Construction

**Comsats Institute of Information Technology**

**Attock Campus**

**Question-1**

In C#, the regular expression library is provided through the System.Text.RegularExpressions namespace. It allows developers to work with regular expressions for pattern matching and manipulation of text. Key components of the library include:

**Regex Class:** The central class for working with regular expressions. It provides methods for pattern matching, replacing, and splitting strings based on regular expressions.

**Match Class:** Represents a single match in the input string. It provides access to captured groups and other match information.

**MatchCollection Class:** Represents a collection of Match objects. It is returned by methods like Regex.Matches().

**Group Class:** Represents a captured group within a match. It allows accessing the value and position of the captured group.

**RegexOptions Enumeration:** Enumerates various options that can be applied to a regular expression pattern, such as case sensitivity and multi-line mode.

**Regex.Escape Method:** Escapes special characters in a string so that it can be used as a literal within a regular expression pattern.

**Regex.Replace Method:** Replaces all occurrences of a pattern in a string with a specified replacement string.

**Regex.Split Method:** Splits an input string into an array of substrings based on a specified regular expression pattern.

**Question-2**

**Right recursive version is:**

S -> c

X -> Y X'

X' -> % Y X' | ε

Y -> Z Y'

Y' -> & Z Y' | ε

Z -> k X k | g

**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;

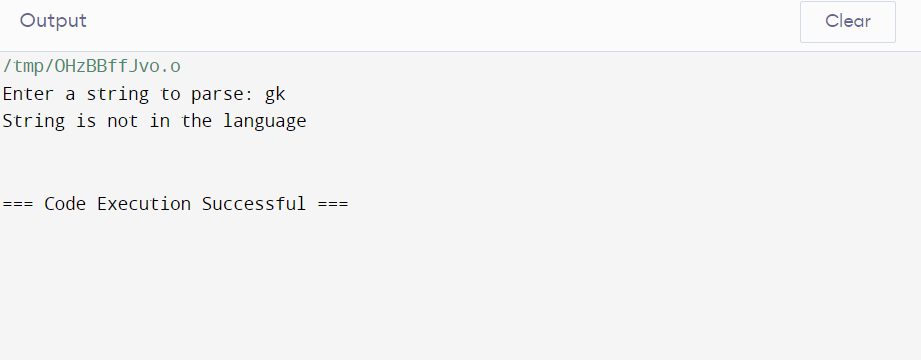
}

return 0;

}

**Output:**

****

****

**Question-3**

**Code: -**

using System;

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;

}

}

**Output: -**

