COMPILER CONSTRUCTION (CSC 441)

LAB MID TERM

COMSATS UNIVERSITY ISLAMABAD

ATTOCK CAMPUS

Submitted By

Hammad Naeem

Registration No

SP21-BCS-010

Submitted To

Sir Bilal Bukhari

Course Title

Compiler Construction

Date

5-4-2023

**QUESTION NO 1:**

Briefly describe the regex library of C#

**Answer:**

**Regular Expression Patterns:**C# allows you to define regular expression patterns using a variety of syntax elements, such as character classes, quantifiers, groupings, assertions, and more.

**Regex Class:**The Regex class is the main entry point for working with regular expressions in C#. It provides methods for matching, replacing, and splitting strings based on regular expression patterns.When a regex pattern matches a string, it returns a Match object containing information about the match, such as the matched text, the position of the match, and any captured groups.

**Groups and Capturing:**Regular expressions can define groups within patterns, allowing you to extract specific parts of a matched string. Captured groups are accessible through the Groups property of a Match object.

**Replacement Patterns:**The Regex.Replace() method allows you to replace matched substrings with specified replacement patterns, which can include references to captured groups.

**Options and Modifiers:**C# supports various options and modifiers that can be applied to regular expression patterns, such as case insensitivity, multiline mode, and single-line mode.

**Validation:**Regular expressions are commonly used for string validation tasks, such as email validation, input sanitization, and format checking.

**QUESTION NO 2:**

Make recursive descent or LL1 parser or recursive descent parser for the following grammar:

S -> X$

X -> X % Y |Y

Y -> Y & Z |Z

Z -> k X k | g

X = E  
T = Y  
F = Z  
+ = %  
MULTIPLY = $

Right recursive version is:

S -> X$  
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;

}

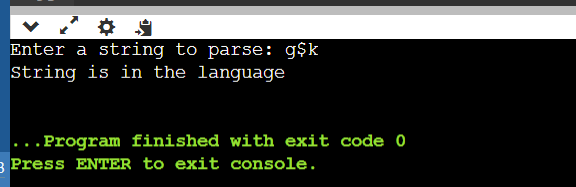
else {

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

}

return 0;}

**Output:**

**Right input: g$k  
**

**Wrong input: kg$k**

**A screen shot of a computer

Description automatically generated**

**QUESTION NO 3:**

Make a Password generator according the following rules:

1. Atleast one uppercase alphabet
2. Atleast 4 numbers , two numbers must be your registration numbers
3. Atleast 2 special characters
4. Must contain initials of first and last name
5. Must contain all odd letters of your first name.
6. Must contain all even letters of your last name.
7. maximum length of 16

**ANSWER:**

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);

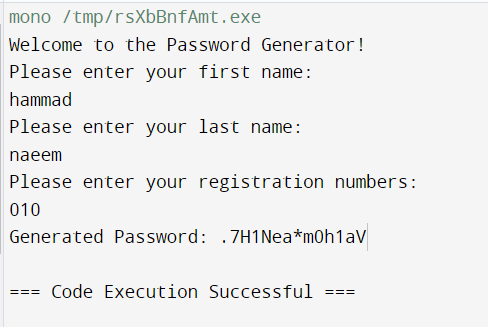
}

return shuffledPassword;

}

}

**Output:**

****