**Lab Mid**

**COMSATS UNIVERSITY**



**ATTOCK CAMPUS**

**Submitted By**

SAFWANA

**Registration No**

SP21-BCS-028

**Submitted To**

Sir Bilal Bukhari

**Date**

05-04-2024

COMPILER CONSTRUCTION (CSC 441)

LAB MID TERM

**QUESTION NO 1:**

Briefly describe the regex library of C#

**Regex Class:** This is the main class in the System.Text.RegularExpressions namespace. It represents a compiled regular expression pattern and provides methods for matching, replacing, and splitting strings based on that pattern.

**Regex Options:** The RegexOptions enum provides various options that can be used to modify the behavior of regular expressions, such as case-insensitive matching, multiline mode, etc.

**Regular Expression Patterns**: Regular expressions in C# are specified using a pattern syntax that allows you to define complex search patterns using metacharacters, quantifiers, character classes, and more.

**Match Class:** Represents the results of a single successful match operation. It provides properties and methods to access the matched text, capture groups, and other match-related information.

**MatchCollection Class**: Represents a collection of Match objects returned by a regex search operation. It provides methods to iterate over the matches and access their properties.

**Group Class:** Represents a capturing group within a regular expression pattern. It provides properties to access the captured text and the position of the group within the overall match.

**Regex Methods:** The Regex class provides static methods like Match, Matches, Replace, and Split for performing various regex operations on strings.

**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

**Code**

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

// Function prototypes

bool parse\_S(char\*);

bool parse\_X(char\*);

bool parse\_X\_prime(char\*);

bool parse\_Y(char\*);

bool parse\_Y\_prime(char\*);

bool parse\_Z(char\*);

// Function to parse S

bool parse\_S(char\* input\_string) {

if (input\_string[0] == 'X' && input\_string[1] == '$')

return true;

else

return false;

}

// Function to parse X

bool parse\_X(char\* input\_string) {

if (parse\_Y(input\_string)) {

char\* remaining\_string = input\_string + 1;

return parse\_X\_prime(remaining\_string);

} else {

return false;

}

}

// Function to parse X'

bool parse\_X\_prime(char\* input\_string) {

if (\*input\_string == '\0')

return true;

else if (\*input\_string == '%' && parse\_Y(input\_string + 1)) {

return parse\_X\_prime(input\_string + 2);

} else {

return false;

}

}

// Function to parse Y

bool parse\_Y(char\* input\_string) {

if (parse\_Z(input\_string)) {

char\* remaining\_string = input\_string + 1;

return parse\_Y\_prime(remaining\_string);

} else {

return false;

}

}

// Function to parse Y'

bool parse\_Y\_prime(char\* input\_string) {

if (\*input\_string == '\0')

return true;

else if (\*input\_string == '&' && parse\_Z(input\_string + 1)) {

return parse\_Y\_prime(input\_string + 2);

} else {

return false;

}

}

// Function to parse Z

bool parse\_Z(char\* input\_string) {

if (\*input\_string == 'k') {

char\* remaining\_string = input\_string + 1;

if (parse\_X(remaining\_string) && input\_string[strlen(input\_string) - 1] == 'k')

return true;

} else if (\*input\_string == 'g') {

return true;

}

return false;

}

int main() {

char input\_string[100];

printf("Enter an input string: ");

scanf("%s", input\_string);

if (parse\_S(input\_string)) {

printf("Input string is accepted.\n");

} else {

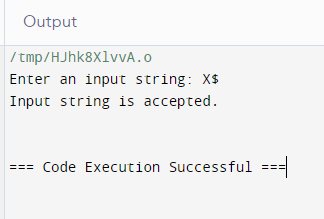
printf("Input string is not accepted.\n");

}

return 0;

}

**Output:**

****

**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

**Code**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include <time.h>

char\* generate\_password(const char\* first\_name, const char\* last\_name, const char\* registration\_numbers) {

char uppercase\_alphabet;

char numbers[5];

char special\_characters[3];

char initials[3];

char odd\_letters\_first\_name[50];

char even\_letters\_last\_name[50];

// Rule (d) - Initials of first and last name

initials[0] = toupper(first\_name[0]);

initials[1] = toupper(last\_name[0]);

initials[2] = '\0';

// Rule (e) - All odd letters of the first name

int j = 0;

for (int i = 0; first\_name[i] != '\0'; i++) {

if (i % 2 == 0) {

odd\_letters\_first\_name[j++] = first\_name[i];

}

}

odd\_letters\_first\_name[j] = '\0';

// Rule (f) - All even letters of the last name

j = 0;

for (int i = 1; last\_name[i] != '\0'; i += 2) {

even\_letters\_last\_name[j++] = last\_name[i];

}

even\_letters\_last\_name[j] = '\0';

// Rule (a) - At least one uppercase alphabet

uppercase\_alphabet = 'A' + rand() % 26;

// Rule (b) - At least 4 numbers, including two registration numbers

numbers[0] = registration\_numbers[0];

numbers[1] = registration\_numbers[1];

numbers[2] = '0' + rand() % 10; // Random number

numbers[3] = registration\_numbers[2];

numbers[4] = '0' + rand() % 10; // Random number

numbers[5] = '\0';

// Rule (c) - At least 2 special characters

special\_characters[0] = '!' + rand() % (126 - '!'); // ASCII range for special characters

special\_characters[1] = '!' + rand() % (126 - '!'); // ASCII range for special characters

special\_characters[2] = '\0';

// Concatenate all components to form the password

char\* password = (char\*)malloc(sizeof(char) \* 18); // 16 characters + null terminator

sprintf(password, "%c%s%s%s%s%s", uppercase\_alphabet, numbers, special\_characters, initials, odd\_letters\_first\_name, even\_letters\_last\_name);

// Shuffle the password to ensure randomness

int length = strlen(password);

for (int i = 0; i < length; i++) {

int j = rand() % length;

char temp = password[i];

password[i] = password[j];

password[j] = temp;

}

return password;

}

int main() {

srand(time(NULL)); // Seed for randomization

const char\* first\_name = "John";

const char\* last\_name = "Doe";

const char\* registration\_numbers = "123456";

char\* password = generate\_password(first\_name, last\_name, registration\_numbers);

printf("Generated password: %s\n", password);

free(password); // Free allocated memory

return 0;

}

