COMSATS University Islamabad,

Attock Campus

**Department of Computer Science**



Submitted by:

**Muhammad Hamza (FA21-BCS-058)**

Submitted to:

**DR. Bilal Bukhari**

**Subject:** Compiler Construction

**Lab Terminal**

**Q6:- Write a C# program using Regular Expressions (Regex) to perform the following tasks:**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text.RegularExpressions;

class Program

{

static void Main(string[] args)

{

// Get the usernames input

Console.Write("Enter usernames (separated by commas): ");

string input = Console.ReadLine();

string[] usernames = input.Split(',');

List<string> validUsernames = new List<string>();

List<string> invalidUsernames = new List<string>();

List<string> result = new List<string>();

foreach (var username in usernames)

{

string trimmedUsername = username.Trim();

string validationResult = ValidateUsername(trimmedUsername);

if (validationResult == "Valid")

{

validUsernames.Add(trimmedUsername);

string password = GeneratePassword();

string passwordStrength = CheckPasswordStrength(password);

result.Add($"{trimmedUsername} - Valid\nGenerated Password: {password} (Strength: {passwordStrength})\n");

}

else

{

invalidUsernames.Add(trimmedUsername);

result.Add($"{trimmedUsername} - Invalid ({validationResult})\n");

}

}

// Display the results to the console

Console.WriteLine("\nValidation Results:");

foreach (var res in result)

{

Console.WriteLine(res);

}

Console.WriteLine($"\nSummary:");

Console.WriteLine($"- Total Usernames: {usernames.Length}");

Console.WriteLine($"- Valid Usernames: {validUsernames.Count}");

Console.WriteLine($"- Invalid Usernames: {invalidUsernames.Count}");

// Write the results to a file

WriteResultsToFile(validUsernames, invalidUsernames, result);

// Retry invalid usernames

if (invalidUsernames.Count > 0)

{

Console.Write("\nDo you want to retry invalid usernames? (y/n): ");

string retryChoice = Console.ReadLine().ToLower();

if (retryChoice == "y")

{

Console.Write("Enter invalid usernames: ");

string retryInput = Console.ReadLine();

string[] retryUsernames = retryInput.Split(',');

foreach (var retry in retryUsernames)

{

string trimmedUsername = retry.Trim();

string retryValidation = ValidateUsername(trimmedUsername);

if (retryValidation == "Valid")

{

validUsernames.Add(trimmedUsername);

string password = GeneratePassword();

string passwordStrength = CheckPasswordStrength(password);

result.Add($"{trimmedUsername} - Valid\nGenerated Password: {password} (Strength: {passwordStrength})\n");

}

else

{

invalidUsernames.Add(trimmedUsername);

result.Add($"{trimmedUsername} - Invalid ({retryValidation})\n");

}

}

Console.WriteLine("\nUpdated Validation Results:");

foreach (var res in result)

{

Console.WriteLine(res);

}

WriteResultsToFile(validUsernames, invalidUsernames, result);

}

}

Console.WriteLine("\nProcessing complete.");

}

// Method to validate the username

static string ValidateUsername(string username)

{

// Check if the username matches the regex

string pattern = @"^[a-zA-Z][a-zA-Z0-9\_]{4,14}$";

if (!Regex.IsMatch(username, pattern))

{

if (username.Length < 5 || username.Length > 15)

return "Username length must be between 5 and 15";

if (!Char.IsLetter(username[0]))

return "Username must start with a letter";

if (username.Any(c => !Char.IsLetterOrDigit(c) && c != '\_'))

return "Username can only contain letters, digits, and underscores";

return "Invalid username format";

}

// Count characters

int upperCount = username.Count(c => Char.IsUpper(c));

int lowerCount = username.Count(c => Char.IsLower(c));

int digitCount = username.Count(c => Char.IsDigit(c));

int underscoreCount = username.Count(c => c == '\_');

// Output the validation details

Console.WriteLine($"{username} - Valid");

Console.WriteLine($"Letters: {upperCount + lowerCount} (Uppercase: {upperCount}, Lowercase: {lowerCount}), Digits: {digitCount}, Underscores: {underscoreCount}");

return "Valid";

}

// Method to generate a secure random password

static string GeneratePassword()

{

Random rand = new Random();

List<char> password = new List<char>();

// Ensure at least 2 uppercase letters, 2 lowercase letters, 2 digits, 2 special characters

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

{

password.Add((char)rand.Next(65, 91)); // Uppercase letter

password.Add((char)rand.Next(97, 123)); // Lowercase letter

password.Add((char)rand.Next(48, 58)); // Digit

password.Add((char)rand.Next(33, 48)); // Special character

}

// Fill the remaining 4 positions with random characters

for (int i = 0; i < 4; i++)

{

char randChar = (char)rand.Next(33, 123); // Random character between '!' and 'z'

password.Add(randChar);

}

// Shuffle the characters to make it random

password = password.OrderBy(x => rand.Next()).ToList();

return new string(password.ToArray());

}

// Method to check the password strength

static string CheckPasswordStrength(string password)

{

int upperCount = password.Count(c => Char.IsUpper(c));

int lowerCount = password.Count(c => Char.IsLower(c));

int digitCount = password.Count(c => Char.IsDigit(c));

int specialCount = password.Count(c => "!@#$%^&\*()\_-+=<>?/.,:;".Contains(c));

// Password Strength Logic

if (password.Length >= 12 && upperCount >= 2 && lowerCount >= 2 && digitCount >= 2 && specialCount >= 2)

return "Strong";

else if (password.Length >= 8 && upperCount >= 1 && lowerCount >= 1 && digitCount >= 1 && specialCount >= 1)

return "Medium";

else

return "Weak";

}

// Method to write results to a file

static void WriteResultsToFile(List<string> validUsernames, List<string> invalidUsernames, List<string> result)

{

string filePath = "UserDetails.txt";

using (StreamWriter writer = new StreamWriter(filePath))

{

writer.WriteLine("Validation Results:");

foreach (var res in result)

{

writer.WriteLine(res);

}

writer.WriteLine($"\nSummary:");

writer.WriteLine($"- Total Usernames: {validUsernames.Count + invalidUsernames.Count}");

writer.WriteLine($"- Valid Usernames: {validUsernames.Count}");

writer.WriteLine($"- Invalid Usernames: {invalidUsernames.Count}");

}

Console.WriteLine($"\nResults saved to {filePath}");

}

}