COMSATS UNIVERSITY

ATTOCK CAMPUS

NAME: SARA BIBI

REG# (SP21-BCS-033)

SUBMITTED TO:

SIR BILAL HAIDER

DATE: 03,JAN , 2025

Part 1: Username Validation

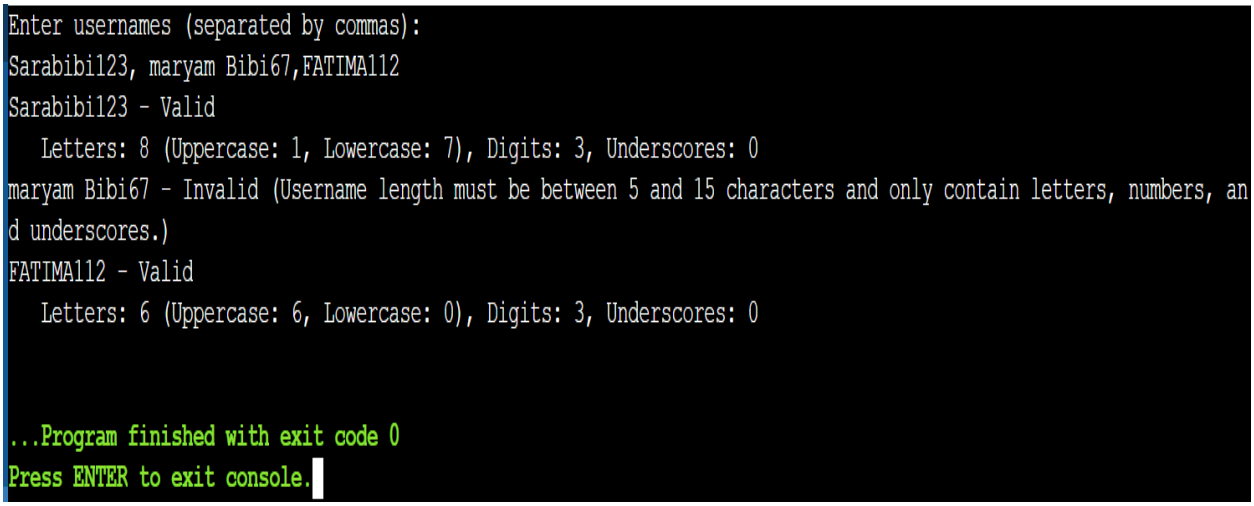
CODE:

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;

class UsernameValidation
{
    static void Main()
    {
        Console.WriteLine("Enter usernames (separated by commas):");
        string input = Console.ReadLine();
        string[] usernames = input.Split(',');

        foreach (string username in usernames)
        {
            string trimmedUsername = username.Trim();
            if (IsValidUsername(trimmedUsername, out string validationResult))
            {
                DisplayUsernameDetails(trimmedUsername);
            }
            else
            {
                Console.WriteLine($"{trimmedUsername} - Invalid ({validationResult})");
            }
        }
    }
```

```
    static bool IsValidUsername(string username, out string validationResult)
    {
        validationResult = string.Empty;

        if (!Regex.IsMatch(username, @"^[a-zA-Z]"))
        {
            validationResult = "Username must start with a letter.";
            return false;
        }
        if (!Regex.IsMatch(username, @"^[a-zA-Z0-9_]{5,15}$"))
        {
            validationResult = "Username length must be between 5 and 15 characters and only
contain letters, numbers, and underscores.";
            return false;
        }

        return true;
    }

    static void DisplayUsernameDetails(string username)
    {
        int upperCount = Regex.Matches(username, "[A-Z]").Count;
        int lowerCount = Regex.Matches(username, "[a-z]").Count;
        int digitCount = Regex.Matches(username, "[0-9]").Count;
        int underscoreCount = Regex.Matches(username, "_").Count;

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

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

OUTPUT:



Part 2: Password Generation:

CODE:

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;


class PasswordGeneration
{
    static void Main()
    {
        Console.WriteLine("Enter usernames (separated by commas):");
        string input = Console.ReadLine();
        string[] usernames = input.Split(',');
```

```
    foreach (string username in usernames)
    {
        string trimmedUsername = username.Trim();
        if (IsValidUsername(trimmedUsername, out string validationResult))
        {
            DisplayUsernameDetails(trimmedUsername);
            string password = "YourPredefinedPassword"; // Set your predefined password here
            string strength = GetPasswordStrength(password);
            Console.WriteLine($"{trimmedUsername}: {password} (Strength: {strength})");
        }
        else
        {
            Console.WriteLine($"{trimmedUsername} - Invalid ({validationResult})");
        }
    }
}

static bool IsValidUsername(string username, out string validationResult)
{
    validationResult = string.Empty;

    if (!Regex.IsMatch(username, @"^[a-zA-Z]"))
    {
        validationResult = "Username must start with a letter.";
        return false;
    }
    if (!Regex.IsMatch(username, @"^[a-zA-Z0-9_]{5,15}$"))
```

```
        {
            validationResult = "Username length must be between 5 and 15 characters and only contain letters, numbers, and underscores.";
            return false;
        }

        return true;
    }

    static void DisplayUsernameDetails(string username)
    {
        int upperCount = Regex.Matches(username, "[A-Z]").Count;
        int lowerCount = Regex.Matches(username, "[a-z]").Count;
        int digitCount = Regex.Matches(username, "[0-9]").Count;
        int underscoreCount = Regex.Matches(username, "_").Count;

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

    static string GetPasswordStrength(string password)
    {
        if (password.Length >= 12 &&
            Regex.IsMatch(password, "[A-Z]") &&
            Regex.IsMatch(password, "[a-z]") &&
            Regex.IsMatch(password, "[0-9]") &&
            Regex.IsMatch(password, "[!@#$%^&*]"))
        {
```

```
            return "Strong";
        }
        return "Medium";
    }
}
```

OUTPUT:

```
Enter usernames (separated by commas):
Sarabibi123,maryam Bibi67,FATIMA112
Sarabibi123 - Valid
   Letters: 8 (Uppercase: 1, Lowercase: 7), Digits: 3, Underscores: 0
Sarabibi123: sarabibi123@$s (Strength: Medium)
maryam Bibi67 - Invalid (Username length must be between 5 and 15 characters and only contain letters, numbers, an
d underscores.)
FATIMA112 - Valid
   Letters: 6 (Uppercase: 6, Lowercase: 0), Digits: 3, Underscores: 0
FATIMA112: sarabibi123@$s (Strength: Medium)


...Program finished with exit code 0
Press ENTER to exit console.
```

Part 3: Enhanced Features:

CODE:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.RegularExpressions;


class Program
{
    static void Main()
    {
        Console.WriteLine("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>();

foreach (string username in usernames)
{
    string trimmedUsername = username.Trim();
    if (IsValidUsername(trimmedUsername, out string validationResult))
    {
        validUsernames.Add(trimmedUsername);
        DisplayUsernameDetails(trimmedUsername);
        string password = GeneratePassword();
        string strength = GetPasswordStrength(password);
        Console.WriteLine($"{trimmedUsername}: {password} (Strength: {strength})");
    }
    else
    {
        invalidUsernames.Add(trimmedUsername);
        Console.WriteLine($"{trimmedUsername} - Invalid ({validationResult})");
    }
}

SaveResultsToFile(usernames, validUsernames, invalidUsernames);

if (invalidUsernames.Count > 0)
{
    Console.WriteLine("\nInvalid Usernames: " + string.Join(", ", invalidUsernames));
```

```
        Console.Write("Do you want to retry invalid usernames? (y/n): ");
        if (Console.ReadLine().Trim().ToLower() == "y")
        {
            Console.Write("Enter invalid usernames: ");
            string retryInput = Console.ReadLine();
            string[] retryUsernames = retryInput.Split(',');

            foreach (string username in retryUsernames)
            {
                string trimmedUsername = username.Trim();
                if (IsValidUsername(trimmedUsername, out string validationResult))
                {
                    validUsernames.Add(trimmedUsername);
                    DisplayUsernameDetails(trimmedUsername);
                    string password = GeneratePassword();
                    string strength = GetPasswordStrength(password);
                    Console.WriteLine($"{trimmedUsername}: {password} (Strength: {strength})");
                }
                else
                {
                    Console.WriteLine($"{trimmedUsername} - Invalid ({validationResult})");
                }
            }
        }

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

```
    static bool IsValidUsername(string username, out string validationResult)
    {
        validationResult = string.Empty;

        if (!Regex.IsMatch(username, @"^[a-zA-Z]"))
        {
            validationResult = "Username must start with a letter.";
            return false;
        }
        if (!Regex.IsMatch(username, @"^[a-zA-Z0-9_]{5,15}$"))
        {
            validationResult = "Username length must be between 5 and 15 characters and only
contain letters, numbers, and underscores.";
            return false;
        }

        return true;
    }

    static void DisplayUsernameDetails(string username)
    {
        int upperCount = Regex.Matches(username, "[A-Z]").Count;
        int lowerCount = Regex.Matches(username, "[a-z]").Count;
        int digitCount = Regex.Matches(username, "[0-9]").Count;
        int underscoreCount = Regex.Matches(username, "_").Count;

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

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

  static string GeneratePassword()
  {
    string upperChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    string lowerChars = "abcdefghijklmnopqrstuvwxyz";
    string digits = "0123456789";
    string specialChars = "!@#$%^&*";
    Random random = new Random();

    List<char> password = new List<char>
    {
      upperChars[random.Next(upperChars.Length)],
      upperChars[random.Next(upperChars.Length)],
      lowerChars[random.Next(lowerChars.Length)],
      lowerChars[random.Next(lowerChars.Length)],
      digits[random.Next(digits.Length)],
      digits[random.Next(digits.Length)],
      specialChars[random.Next(specialChars.Length)],
      specialChars[random.Next(specialChars.Length)]
    };

    while (password.Count < 12)
    {
      string allChars = upperChars + lowerChars + digits + specialChars;
      password.Add(allChars[random.Next(allChars.Length)]);
```

```
        }

        password.Sort((x, y) => random.Next(-1, 2));
        return new string(password.ToArray());
    }

    static string GetPasswordStrength(string password)
    {
        if (password.Length >= 12 &&
            Regex.IsMatch(password, "[A-Z]") &&
            Regex.IsMatch(password, "[a-z]") &&
            Regex.IsMatch(password, "[0-9]") &&
            Regex.IsMatch(password, "[!@#$%^&*]"))
        {
            return "Strong";
        }
        return "Medium";
    }

    static void SaveResultsToFile(string[] usernames, List<string> validUsernames, List<string>
invalidUsernames)
    {
        using (StreamWriter writer = new StreamWriter("UserDetails.txt"))
        {
            writer.WriteLine("Validation Results:");
            foreach (string username in usernames)
            {
                string trimmedUsername = username.Trim();
```

```
            if (validUsernames.Contains(trimmedUsername))
            {
                int upperCount = Regex.Matches(trimmedUsername, "[A-Z]").Count;
                int lowerCount = Regex.Matches(trimmedUsername, "[a-z]").Count;
                int digitCount = Regex.Matches(trimmedUsername, "[0-9]").Count;
                int underscoreCount = Regex.Matches(trimmedUsername, "_").Count;

                string password = GeneratePassword();
                string strength = GetPasswordStrength(password);

                writer.WriteLine($"{trimmedUsername} - Valid");
                writer.WriteLine($"   Letters: {upperCount + lowerCount} (Uppercase: {upperCount}, Lowercase: {lowerCount}), Digits: {digitCount}, Underscores: {underscoreCount}");
                writer.WriteLine($"   Generated Password: {password} (Strength: {strength})");
            }
            else
            {
                writer.WriteLine($"{trimmedUsername} - Invalid");
            }
        }

        writer.WriteLine("\nSummary:");
        writer.WriteLine($"- Total Usernames: {usernames.Length}");
        writer.WriteLine($"- Valid Usernames: {validUsernames.Count}");
        writer.WriteLine($"- Invalid Usernames: {invalidUsernames.Count}");
    }
  }
}
```

OUTPUT:

```
Enter usernames (separated by commas):
Sarabibi123,maryam Bibi67,FATIMA112
Sarabibi123 - Valid
   Letters: 8 (Uppercase: 1, Lowercase: 7), Digits: 3, Underscores: 0
Sarabibi123: MTz80l%%G8Hz (Strength: Strong)
maryam Bibi67 - Invalid (Username length must be between 5 and 15 characters and only contain letters, numbers, an
d underscores.)
FATIMA112 - Valid
   Letters: 6 (Uppercase: 6, Lowercase: 0), Digits: 3, Underscores: 0
FATIMA112: INl2o9%bmf*Z (Strength: Strong)

Invalid Usernames: maryam Bibi67
Do you want to retry invalid usernames? (y/n): Y
Enter invalid usernames: maryam Bibi67
maryam Bibi67 - Invalid (Username length must be between 5 and 15 characters and only contain letters, numbers, an
d underscores.)

Processing complete.


...Program finished with exit code 0
Press ENTER to exit console.
```

VALIDATION SUMMARY:

main.cs UserDetails.txt

```
1  Validation Results:
2  Sarabibi123 - Valid
3     Letters: 8 (Uppercase: 1, Lowercase: 7), Digits: 3, Underscores: 0
4     Generated Password: OYv2&4hZ!m7! (Strength: Strong)
5  maryam Bibi67 - Invalid
6  FATIMA112 - Valid
7     Letters: 6 (Uppercase: 6, Lowercase: 0), Digits: 3, Underscores: 0
8     Generated Password: FJd&v7&2%87k (Strength: Strong)
9
10 Summary:
11 - Total Usernames: 3
12 - Valid Usernames: 2
13 - Invalid Usernames: 1
14
```

SNo	Identifier	Scope	Value	Type	Parameter Type (for functions)
1	main	0	0	FUNCTION	INT, FLOAT
2	printf	0	0	FUNCTION	CHAR
3	a	0	10	INT	
4	b	0	20.5	FLOAT	
5	c	0		ARRAY	
6	i	0	5	INT	
7	sum	0	15.2	FLOAT	