

**Lab Mid**

**Name:**

**Kinza noor**

**Reg. #:**

**Fa21-BCS-028**

**Course:**

**Compiler Construction**

**Instructors:**

**Sir Bilal Haider Bukhari**

Q2: Create a password generator using regex with the following specifications (Marks 7.5)

1- Contains 2 digit of your registeration number

2- Contains second letters from your first name and last name

3- Contains 2 characters from your favorite movie

4- Contains special characters

5- Does not contain # sign

6- Length should be 14

**Code**:

using System;

using System.Text;

using System.Text.RegularExpressions;

class PasswordGenerator

{

static void Main()

{

string movieName = "junnoniat"; // Complete movie name

string password = GeneratePassword(movieName);

Console.WriteLine("Generated Password: " + password);

}

static string GeneratePassword(string movieName)

{

string registrationNumberDigits = "28";

string secondLetterFirstName = "i";

string secondLetterLastName = "o";

string specialCharacters = "!@$%^&\*";

Random random = new Random();

StringBuilder passwordBuilder = new StringBuilder();

passwordBuilder.Append(registrationNumberDigits);

passwordBuilder.Append(secondLetterFirstName);

passwordBuilder.Append(secondLetterLastName);

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

{

char randomMovieChar = movieName[random.Next(movieName.Length)];

passwordBuilder.Append(randomMovieChar);

}

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

passwordBuilder.Append(specialCharacters[random.Next(specialCharacters.Length)]);

string allChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789" + specialCharacters;

while (passwordBuilder.Length < 14)

passwordBuilder.Append(allChars[random.Next(allChars.Length)]);

char[] passwordArray = passwordBuilder.ToString().ToCharArray();

Array.Sort(passwordArray, (a, b) => random.Next(-1, 2));

string password = new string(passwordArray);

string pattern = @"^(?!.\*#).{14}$";

if (Regex.IsMatch(password, pattern))

return password;

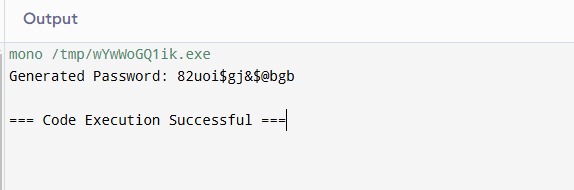
else

return GeneratePassword(movieName);

    }

}

**Output**:



Q3: Ziyad bought a toy remote car, the car has a controller and can pass the following commands start, stop accelerate, right turn. Ziyad was able to find that the car does not turn left. Create a context free grammer to help ziyad sort out the issue with his car

Non Terminals

- Goal Symbol

- CMD

- Action

Terminals

- Start

- Stop

- Accelerate

- Brake

- Left

- Right

**Code**:

using System;

using System.Collections.Generic;

public class RemoteCar

{

private static HashSet<string> validCommands = new HashSet<string>

{

"Start", "Stop", "Accelerate", "Brake", "Right", "Left"

};

public static void Main()

{

Console.WriteLine("Enter commands (separated by commas): ");

string input = Console.ReadLine();

if (input != null)

{

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

foreach (var command in commands)

{

string trimmedCommand = command.Trim();

if (IsValidCommand(trimmedCommand))

{

Console.WriteLine($"Command '{trimmedCommand}' is valid.");

}

else

{

Console.WriteLine($"Command '{trimmedCommand}' is invalid. The car cannot turn left.");

}

}

}

}

private static bool IsValidCommand(string command)

{

if (command == "start" || command == "stop" || command == "accelerate" || command == "brake" || command == "right")

{

return true;

}

else if (command == "left")

{

return false;

}

return false;

    }

}

**Output:**

