Logo, company name

Description automatically generated

**COMSATS UNIVERSITY**

**ATTOCK CAMPUS**

**NAME: SARA BIBI**

**REG# (SP21-BCS-033)**

**SUBMITTED TO:**

**SIR BILAL HAIDER**

**DATE: 25,OCT , 2024**

**QUESTION NO: 02**

**CODE:**

using System;

using System.Text;

using System.Text.RegularExpressions;

class PasswordGenerator

{

static void Main()

{

string password = GeneratePassword();

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

}

static string GeneratePassword()

{

string registrationNumberDigits = "33";

string secondLetterFirstName = "a";

string secondLetterLastName = "i";

string favoriteMovieChars = "jn";

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

Random random = new Random();

StringBuilder passwordBuilder = new StringBuilder();

passwordBuilder.Append(registrationNumberDigits);

passwordBuilder.Append(secondLetterFirstName);

passwordBuilder.Append(secondLetterLastName);

passwordBuilder.Append(favoriteMovieChars);

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;

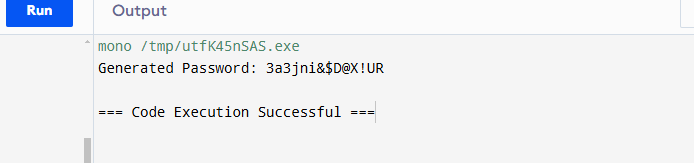
else

return GeneratePassword();

}

}

**OUTPUT:**



**QUESTION NO:03**

**CODE:**

using System;

using System.Text.RegularExpressions;

class CarCommandParser

{

public static void Main()

{

string[] commands = { "Start", "Accelerate", "Right", "Left", "Stop", "Brake" };

foreach (var command in commands)

{

if (ParseCommand(command))

Console.WriteLine($"'{command}' is a valid command.");

else

Console.WriteLine($"'{command}' is NOT a valid command. Left turn is not supported.");

}

}

public static bool ParseCommand(string command)

{

// Context-Free Grammar:

// GoalSymbol -> CMD

// CMD -> Action | CMD Action

// Action -> Start | Stop | Accelerate | Brake | Right

// Note: 'Left' is missing and should not pass

// Regular expressions for each terminal

string pattern = @"^(Start|Stop|Accelerate|Brake|Right)$";

// Check if command matches any valid terminal

return Regex.IsMatch(command, pattern);

}

}

**OUTPUT:**

