Lab Mid Exam

Name: **Sana Saeed**

Reg No: **Sp21-bcs-032**

Course: **Compiler construction**

Date: **06/04/2024**

Submitted To:

Mr. Syed Bilal Haider

QUESTION NO 1:

Briefly describe the regex library of C#

Answer:

The theoretical aspects of regular expressions and how they are implemented in the C# Regex library:

Regular Expressions (Regex):

- Regular expressions are a powerful tool for pattern matching and string manipulation.
- They provide a concise and flexible means for describing text patterns.
- Regex patterns consist of a combination of literal characters, metacharacters, and quantifiers.
- Metacharacters such as ^, $, ., *, +, ?, [,], {, }, (,), \, etc., have special meanings within a regex pattern.
- Quantifiers like *, +, ?, {n}, {n,}, {n,m} specify the number of occurrences of a preceding element.

Regex Class in C#:

- The Regex class in C# encapsulates a compiled representation of a regular expression pattern.
- It provides methods for pattern matching, searching, replacing, and splitting strings based on regex patterns.
- The System.Text.RegularExpressions namespace contains this class and related types.
- The RegexOptions enum allows specifying various options like case insensitivity, multiline mode, and more.
- Example:

  ```
  Regex regex = new Regex(@"\b\d{3}\b", RegexOptions.IgnoreCase);
  ```

Match and MatchCollection:

- The Match class represents a single match of a regex pattern within a string.
- It provides properties to access the matched value and captured groups within the match.
- The MatchCollection class represents a collection of matches found within a string.
- It allows iterating over multiple matches found in a single string.

Groups and Capturing:

- Parentheses () in a regex pattern create capturing groups.
- Capturing groups allow extracting portions of the matched string.
- Each capturing group can be accessed using the Groups property of a Match object.

Constructors:

The Regex class has various constructors for initializing instances with different parameters.

Example:

- Regex regex1 = new Regex(@"\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b", RegexOptions.IgnoreCase);
- Regex regex2 = new Regex(@"\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}\b", RegexOptions.IgnoreCase | RegexOptions.Compiled);

Properties:

- **Options:** Gets the options passed into the Regex constructor.
- **MatchTimeout:** Gets the time-out interval for pattern matching.

Example:

- Regex regex = new Regex(@"\b\d{3}\b", RegexOptions.IgnoreCase);
- RegexOptions options = regex.Options; // Returns RegexOptions.IgnoreCase

Methods:

- **Match:** Searches the specified input string for the first occurrence of the regular expression.
- **Matches:** Searches the specified input string for all occurrences of the regular expression.
- **Replace:** Replaces all occurrences of the regular expression pattern with a specified replacement string.

Example:

string input = "The price is $100 and $200.";

string pattern = @"\$\d+";

string result = Regex.Replace(input, pattern, "[$&]");

// Output: "The price is [$100] and [$200]."

Performance Considerations:

- Regular expressions can be computationally expensive, especially for complex patterns or large input strings.
- Compiling a regex pattern can improve performance for multiple uses of the same pattern.
- The RegexOptions.Compiled option compiles the regex pattern for improved performance.

Use Cases:

- **Validation:** Validate email addresses, phone numbers, etc.
- **Manipulation:** Replace or extract substrings based on patterns.
- **Search and Extraction:** Find and extract specific information from text.

Regular expressions are powerful tools for string manipulation and text processing tasks, and the C# Regex class provides a convenient way to work with them in your applications.

Question No 02:

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

Answer:

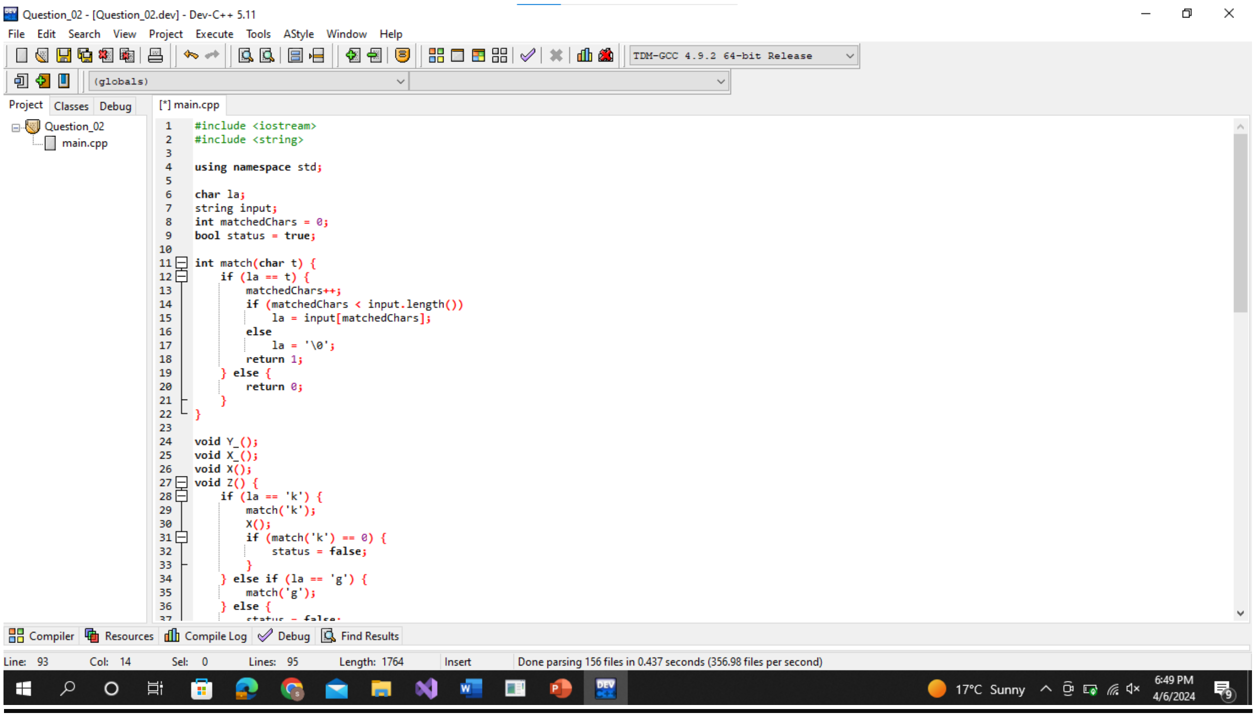
Making the above grammar right recursive to implement recursive descent or LL1 parser or recursive descent parser Hence, resulted grammar is

CFG:

Right Recursive Grammer:

```
S -> X$
X -> YX'
X' -> %YX' | ε
Y -> ZY'
Y' -> &ZY' | ε
 Z -> kXk | g
```

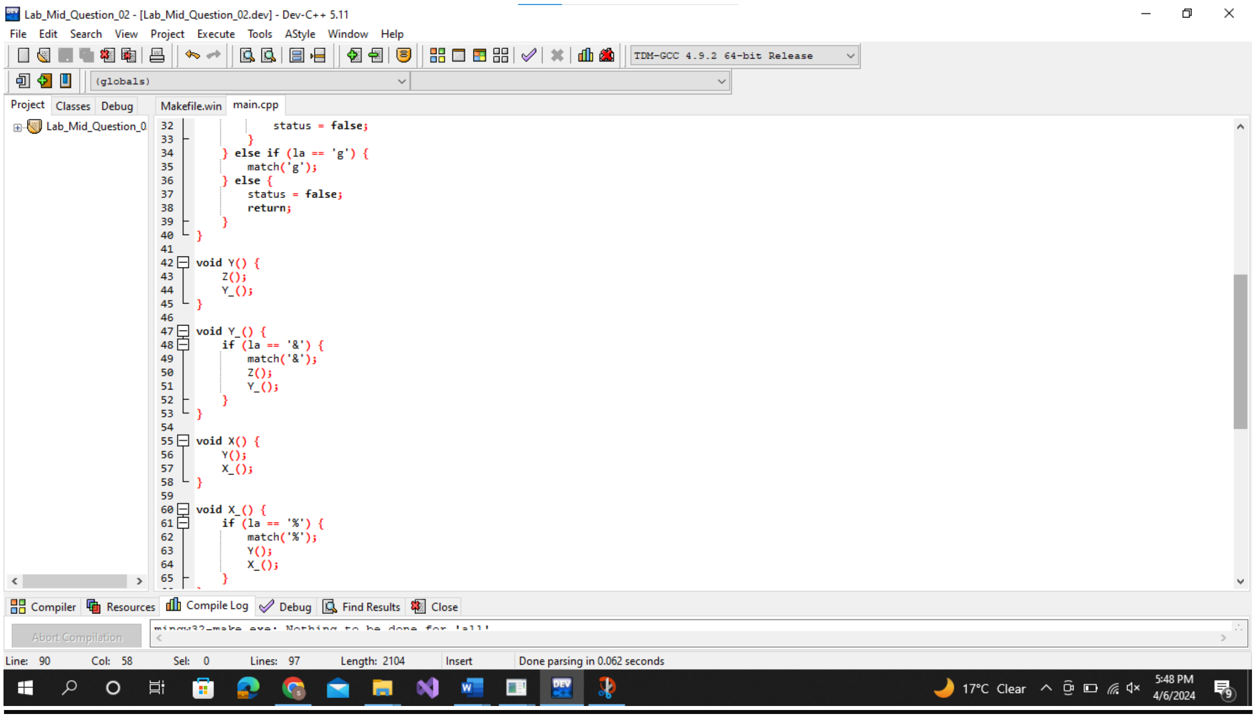
Code:


Question_02 - [Question_02.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
[*] main.cpp
Question_02
main.cpp
Compiler Resources Compile Log Debug Find Results

```cpp
#include <iostream>
#include <string>

using namespace std;

char la;
string input;
int matchedChars = 0;
bool status = true;

int match(char t) {
    if (la == t) {
        matchedChars++;
        if (matchedChars < input.length())
            la = input[matchedChars];
        else
            la = '\0';
        return 1;
    } else {
        return 0;
    }
}

void Y_();
void X_();
void X();
void Z() {
    if (la == 'k') {
        match('k');
        X();
        if (match('k') == 0) {
            status = false;
        }
    } else if (la == 'g') {
        match('g');
    } else {
        status = false;
```

Line: 93 Col: 14 Sel: 0 Lines: 95 Length: 1764 Insert Done parsing 156 files in 0.437 seconds (356.98 files per second)

17°C Sunny
6:49 PM
4/6/2024

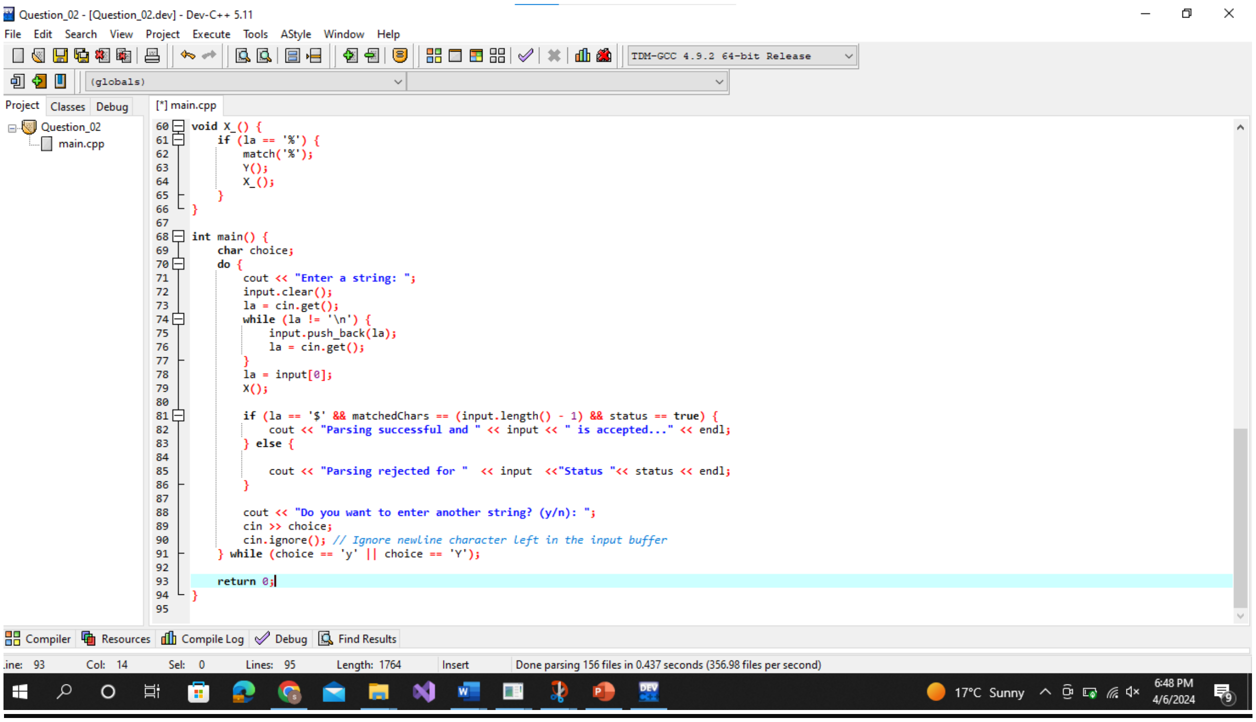

Lab_Mid_Question_02 - [Lab_Mid_Question_02.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Makefile.win main.cpp
Lab_Mid_Question_0
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation

```cpp
            status = false;
        }
    } else if (la == 'g') {
        match('g');
    } else {
        status = false;
        return;
    }
}

void Y() {
    Z();
    Y_();
}

void Y_() {
    if (la == '&') {
        match('&');
        Z();
        Y_();
    }
}

void X() {
    Y();
    X_();
}

void X_() {
    if (la == '%') {
        match('%');
        Y();
        X_();
    }
```

```
mingw32-make.exe: Nothing to be done for 'all'
```

Line: 90 Col: 58 Sel: 0 Lines: 97 Length: 2104 Insert Done parsing in 0.062 seconds

17°C Clear
5:48 PM
4/6/2024

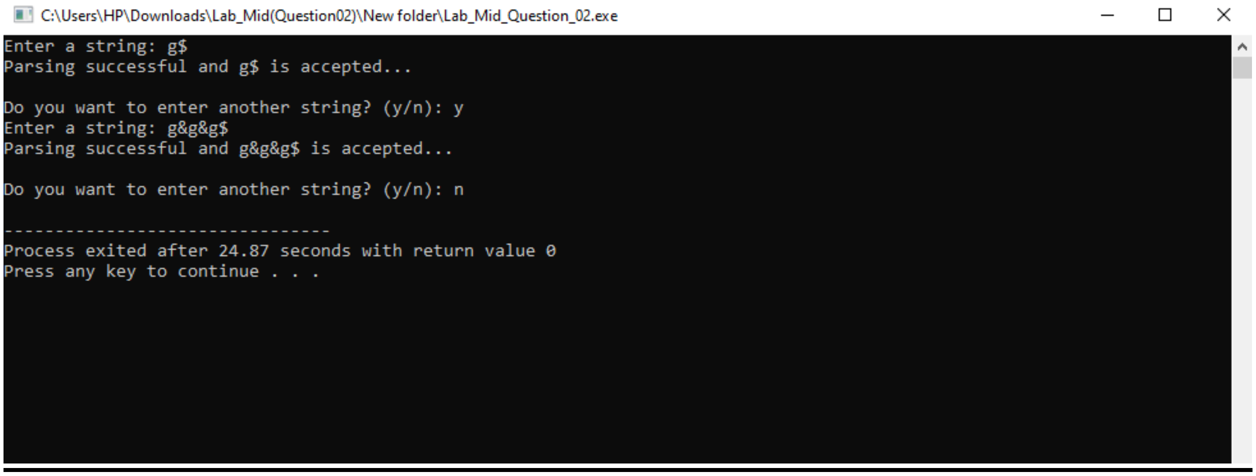

Question_02 - [Question_02.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Question_02
main.cpp
[*] main.cpp
Compiler Resources Compile Log Debug Find Results
Line: 93 Col: 14 Sel: 0 Lines: 95 Length: 1764 Insert Done parsing 156 files in 0.437 seconds (356.98 files per second)
17°C Sunny
6:48 PM
4/6/2024

```
void X_() {
    if (la == '%') {
        match('%');
        Y();
        X_();
    }
}

int main() {
    char choice;
    do {
        cout << "Enter a string: ";
        input.clear();
        la = cin.get();
        while (la != '\n') {
            input.push_back(la);
            la = cin.get();
        }
        la = input[0];
        X();

        if (la == '$' && matchedChars == (input.length() - 1) && status == true) {
            cout << "Parsing successful and " << input << " is accepted..." << endl;
        } else {

            cout << "Parsing rejected for "  << input  <<"Status "<< status << endl;
        }

        cout << "Do you want to enter another string? (y/n): ";
        cin >> choice;
        cin.ignore(); // Ignore newline character left in the input buffer
    } while (choice == 'y' || choice == 'Y');

    return 0;
}
```

Output:

The are the inputs that are checked:

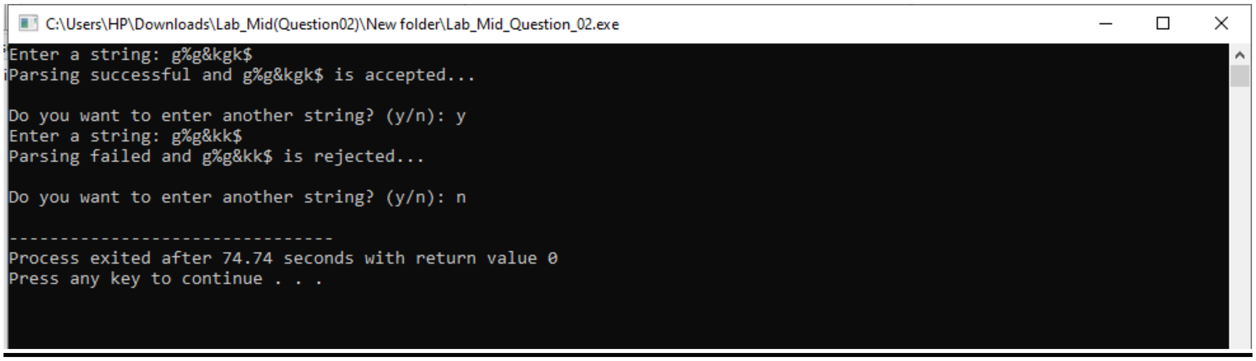
- g$
- g&g&g$


C:\Users\HP\Downloads\Lab_Mid(Question02)\New folder\Lab_Mid_Question_02.exe

```
Enter a string: g$
Parsing successful and g$ is accepted...

Do you want to enter another string? (y/n): y
Enter a string: g&g&g$
Parsing successful and g&g&g$ is accepted...

Do you want to enter another string? (y/n): n

--------------------------------
Process exited after 24.87 seconds with return value 0
Press any key to continue . . .
```

The are the inputs that are checked:

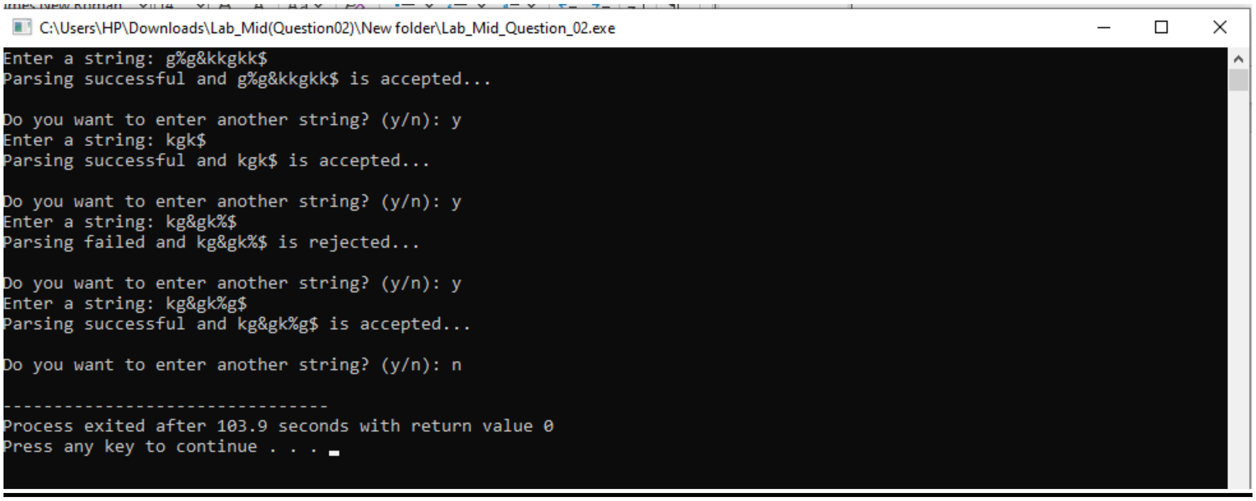
- g%g&kgk$
- g%g&kk$


C:\Users\HP\Downloads\Lab_Mid(Question02)\New folder\Lab_Mid_Question_02.exe

```
Enter a string: g%g&kgk$
Parsing successful and g%g&kgk$ is accepted...

Do you want to enter another string? (y/n): y
Enter a string: g%g&kk$
Parsing failed and g%g&kk$ is rejected...

Do you want to enter another string? (y/n): n

--------------------------------
Process exited after 74.74 seconds with return value 0
Press any key to continue . . .
```

The are the inputs that are checked:

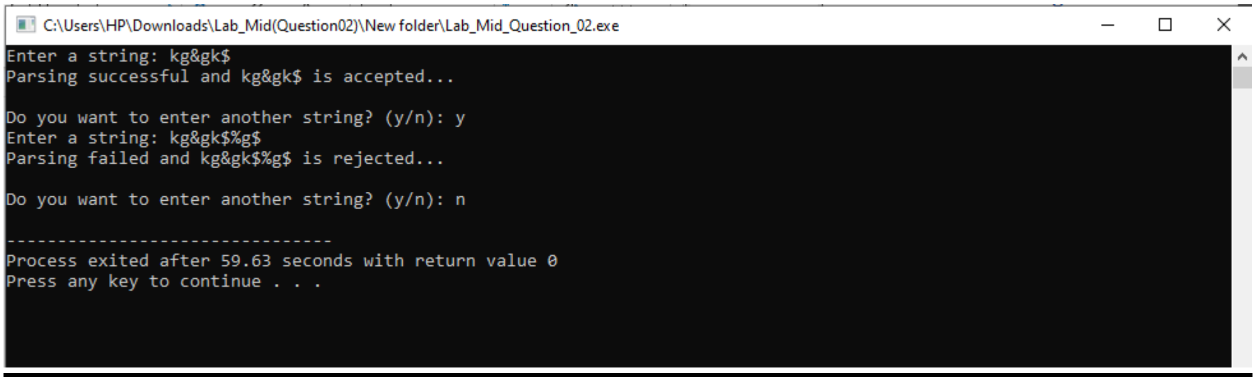
- g%g&kkgkk$
- kgk$
- kg&gk%$
- kg&gk%g$


C:\Users\HP\Downloads\Lab_Mid(Question02)\New folder\Lab_Mid_Question_02.exe

```
Enter a string: g%g&kkgkk$
Parsing successful and g%g&kkgkk$ is accepted...

Do you want to enter another string? (y/n): y
Enter a string: kgk$
Parsing successful and kgk$ is accepted...

Do you want to enter another string? (y/n): y
Enter a string: kg&gk%$
Parsing failed and kg&gk%$ is rejected...

Do you want to enter another string? (y/n): y
Enter a string: kg&gk%g$
Parsing successful and kg&gk%g$ is accepted...

Do you want to enter another string? (y/n): n

--------------------------------
Process exited after 103.9 seconds with return value 0
Press any key to continue . . .
```

The are the inputs that are checked:

- kg&gk$
- kg&gk$%g$


C:\Users\HP\Downloads\Lab_Mid(Question02)\New folder\Lab_Mid_Question_02.exe

```
Enter a string: kg&gk$
Parsing successful and kg&gk$ is accepted...

Do you want to enter another string? (y/n): y
Enter a string: kg&gk$%g$
Parsing failed and kg&gk$%g$ is rejected...

Do you want to enter another string? (y/n): n

--------------------------------
Process exited after 59.63 seconds with return value 0
Press any key to continue . . .
```

Question No 03:

Make a Password generator according the following rules:

(a) Atleast one uppercase alphabet

(b) Atleast 4 numbers , two numbers must be your registration numbers

(c) Atleast 2 special characters

(d) Must contain initials of first and last name

(e) Must contain all odd letters of your first name.

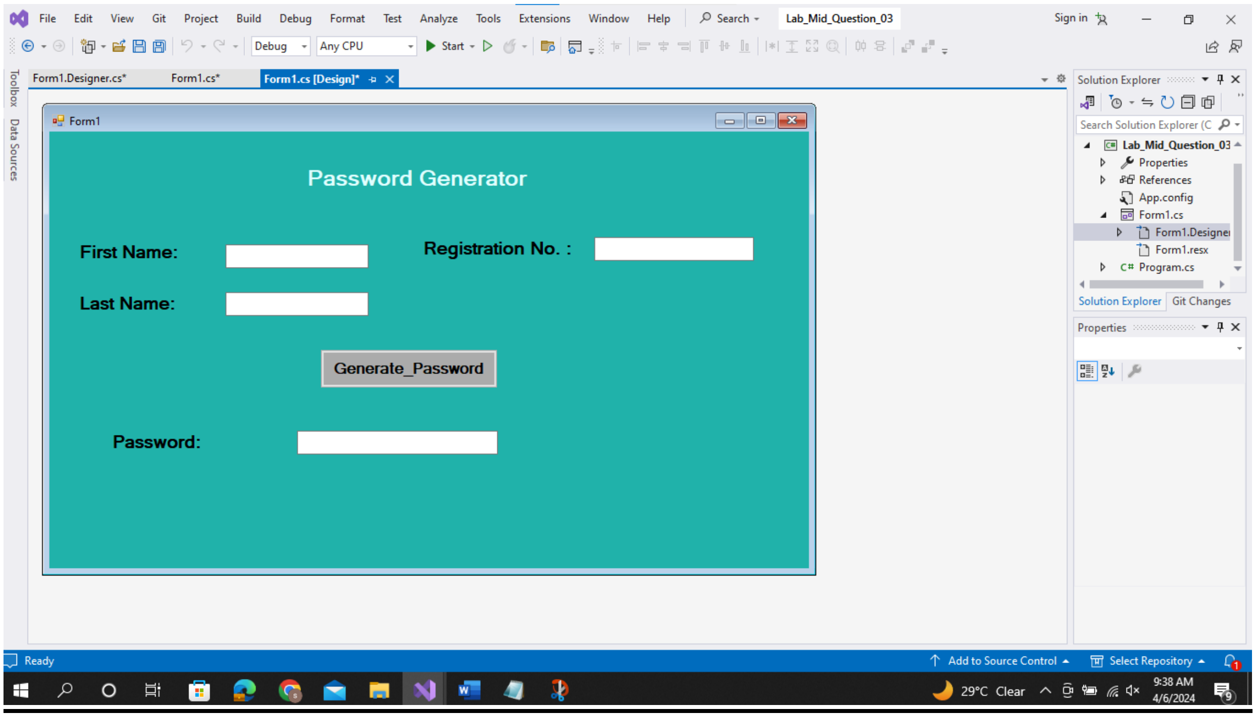
(f) Must contain all even letters of your last name.

(g) maximum length of 16

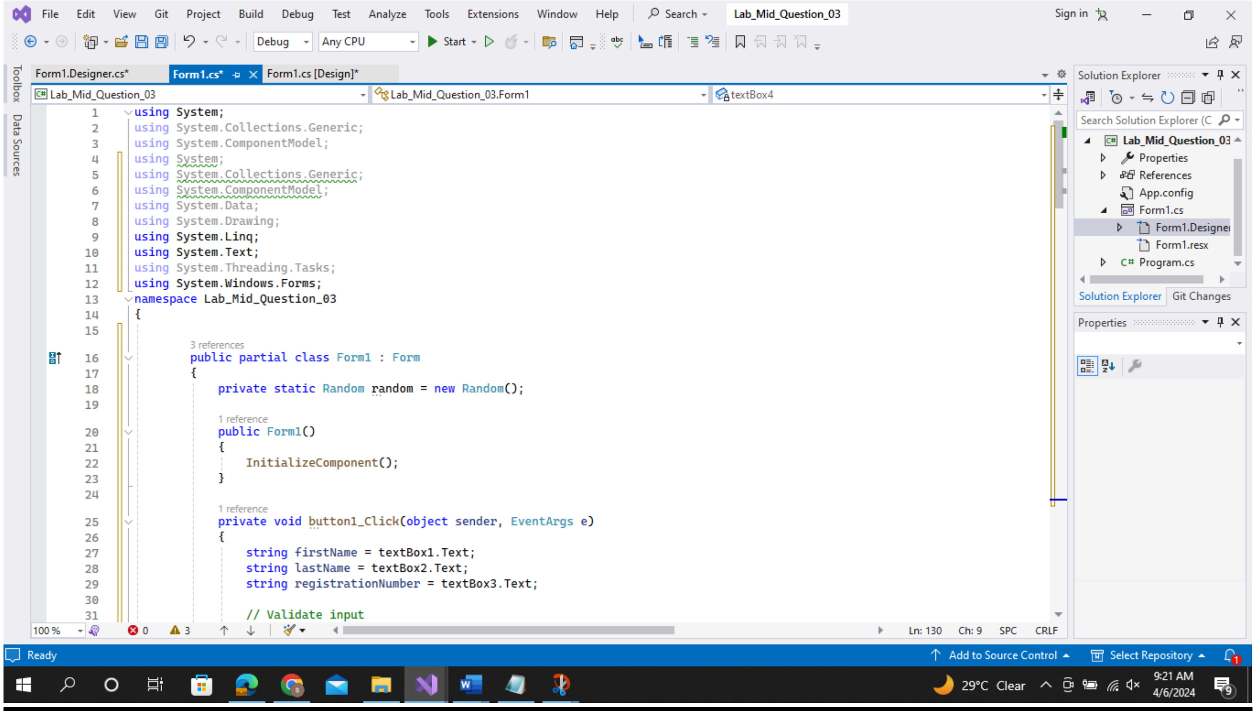
Answer:

It’s a Windows Form Application Hence,

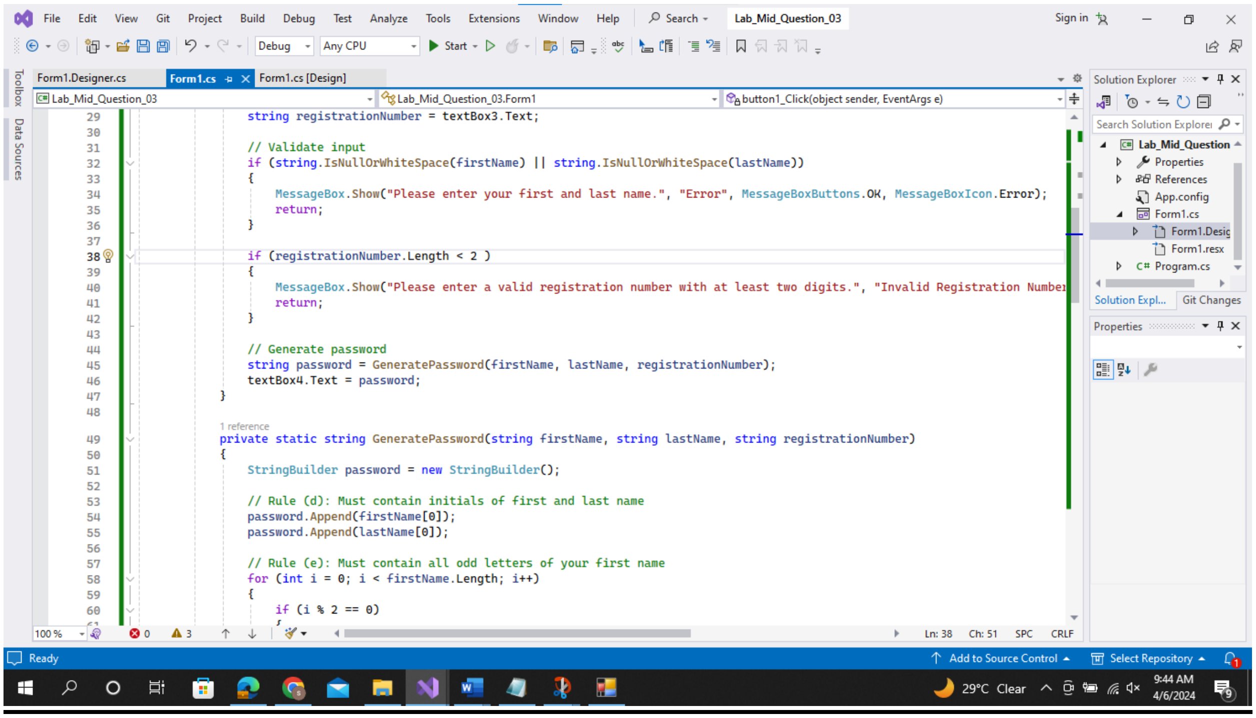
Form(Design).cs:



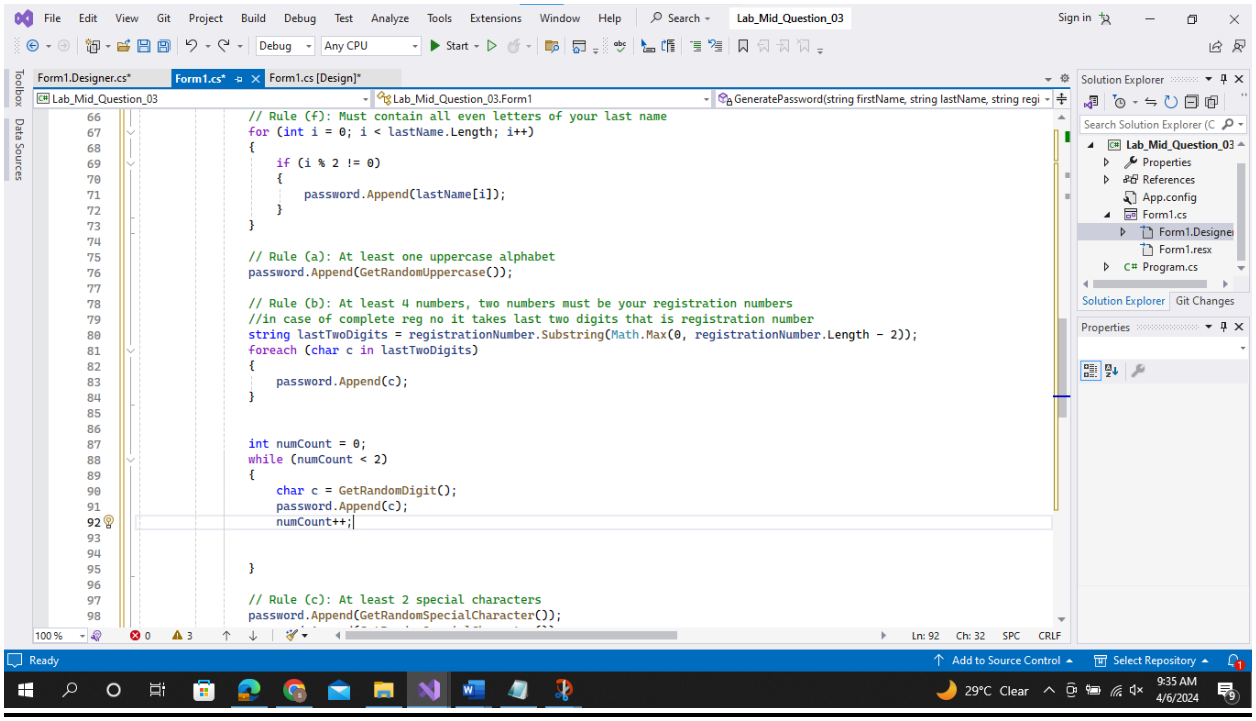
Form.cs:



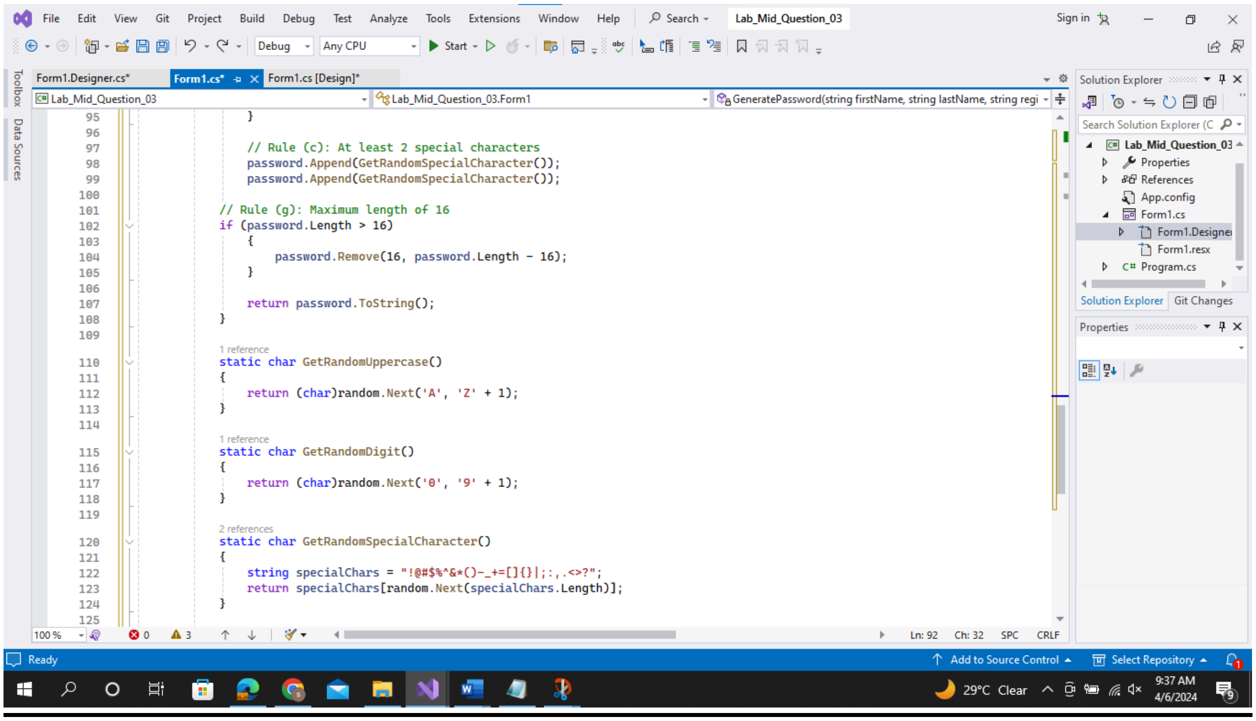
```csharp
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
namespace Lab_Mid_Question_03
{

        3 references
        public partial class Form1 : Form
        {
            private static Random random = new Random();

            1 reference
            public Form1()
            {
                InitializeComponent();
            }

            1 reference
            private void button1_Click(object sender, EventArgs e)
            {
                string firstName = textBox1.Text;
                string lastName = textBox2.Text;
                string registrationNumber = textBox3.Text;

                // Validate input
```



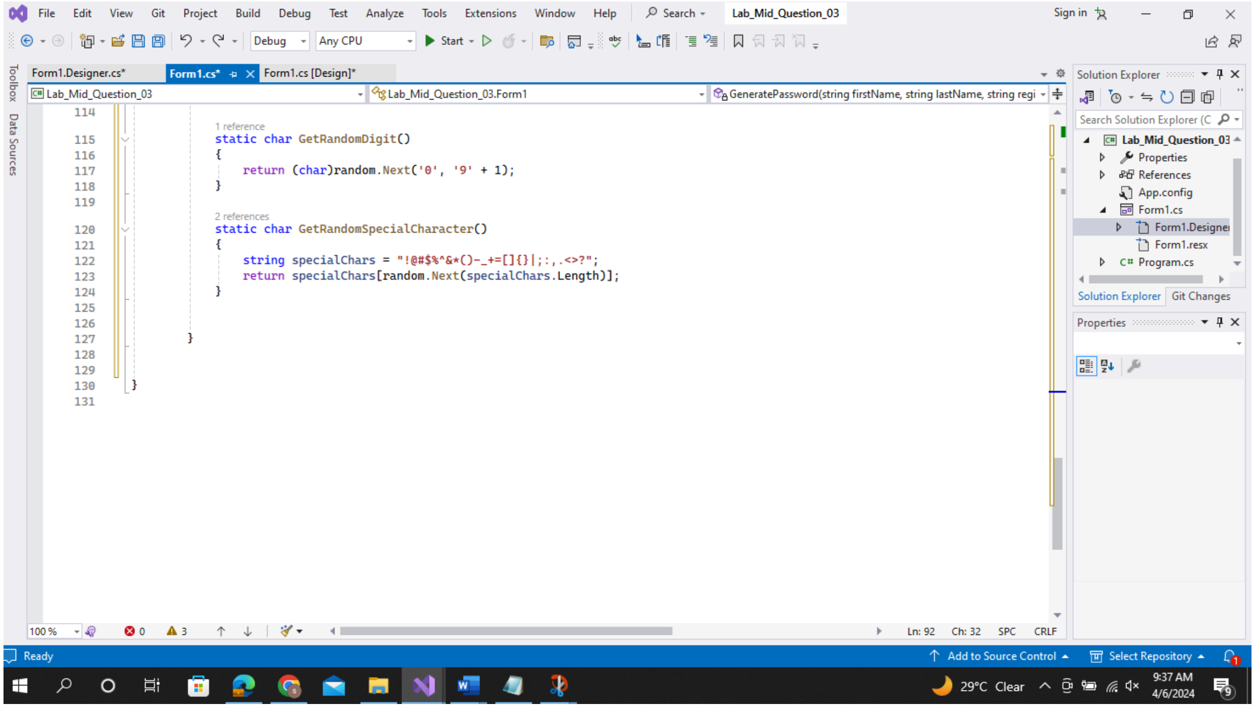
```csharp
                string registrationNumber = textBox3.Text;

                // Validate input
                if (string.IsNullOrWhiteSpace(firstName) || string.IsNullOrWhiteSpace(lastName))
                {
                    MessageBox.Show("Please enter your first and last name.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                if (registrationNumber.Length < 2 )
                {
                    MessageBox.Show("Please enter a valid registration number with at least two digits.", "Invalid Registration Number
                    return;
                }

                // Generate password
                string password = GeneratePassword(firstName, lastName, registrationNumber);
                textBox4.Text = password;
            }

            1 reference
            private static string GeneratePassword(string firstName, string lastName, string registrationNumber)
            {
                StringBuilder password = new StringBuilder();

                // Rule (d): Must contain initials of first and last name
                password.Append(firstName[0]);
                password.Append(lastName[0]);

                // Rule (e): Must contain all odd letters of your first name
                for (int i = 0; i < firstName.Length; i++)
                {
                    if (i % 2 == 0)
```



```
            // Rule (f): Must contain all even letters of your last name
            for (int i = 0; i < lastName.Length; i++)
            {
                if (i % 2 != 0)
                {
                    password.Append(lastName[i]);
                }
            }

            // Rule (a): At least one uppercase alphabet
            password.Append(GetRandomUppercase());

            // Rule (b): At least 4 numbers, two numbers must be your registration numbers
            //in case of complete reg no it takes last two digits that is registration number
            string lastTwoDigits = registrationNumber.Substring(Math.Max(0, registrationNumber.Length - 2));
            foreach (char c in lastTwoDigits)
            {
                password.Append(c);
            }


            int numCount = 0;
            while (numCount < 2)
            {
                char c = GetRandomDigit();
                password.Append(c);
                numCount++;


            }

            // Rule (c): At least 2 special characters
            password.Append(GetRandomSpecialCharacter());
```



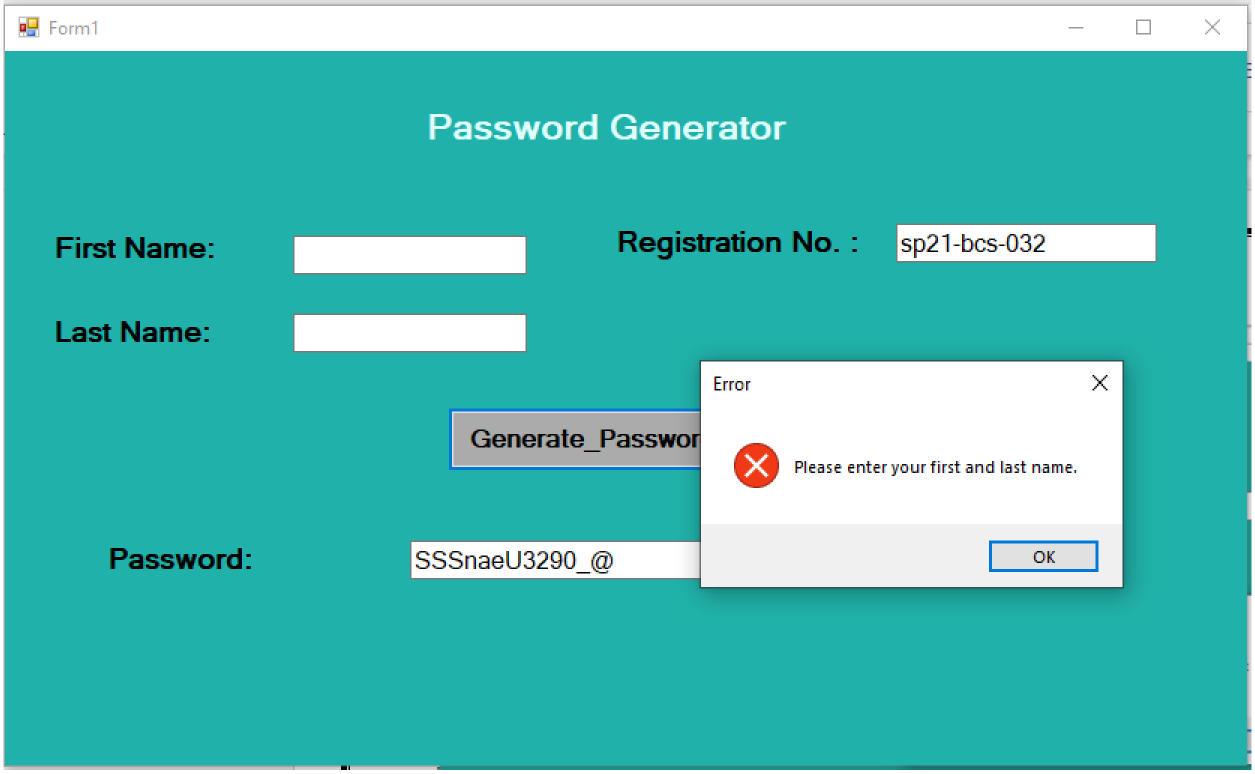
```
            }

            // Rule (c): At least 2 special characters
            password.Append(GetRandomSpecialCharacter());
            password.Append(GetRandomSpecialCharacter());

        // Rule (g): Maximum length of 16
        if (password.Length > 16)
            {
                password.Remove(16, password.Length - 16);
            }

            return password.ToString();
        }

        static char GetRandomUppercase()
        {
            return (char)random.Next('A', 'Z' + 1);
        }

        static char GetRandomDigit()
        {
            return (char)random.Next('0', '9' + 1);
        }

        static char GetRandomSpecialCharacter()
        {
            string specialChars = "!@#$%^&*()-_+=[]{}|;:,.<>?";
            return specialChars[random.Next(specialChars.Length)];
        }
```



```
        1 reference
        static char GetRandomDigit()
        {
            return (char)random.Next('0', '9' + 1);
        }

        2 references
        static char GetRandomSpecialCharacter()
        {
            string specialChars = "!@#$%^&*()-_+=[]{}|;:,.<>?";
            return specialChars[random.Next(specialChars.Length)];
        }


    }

}
```

Output:

Input Validations:



First Name:

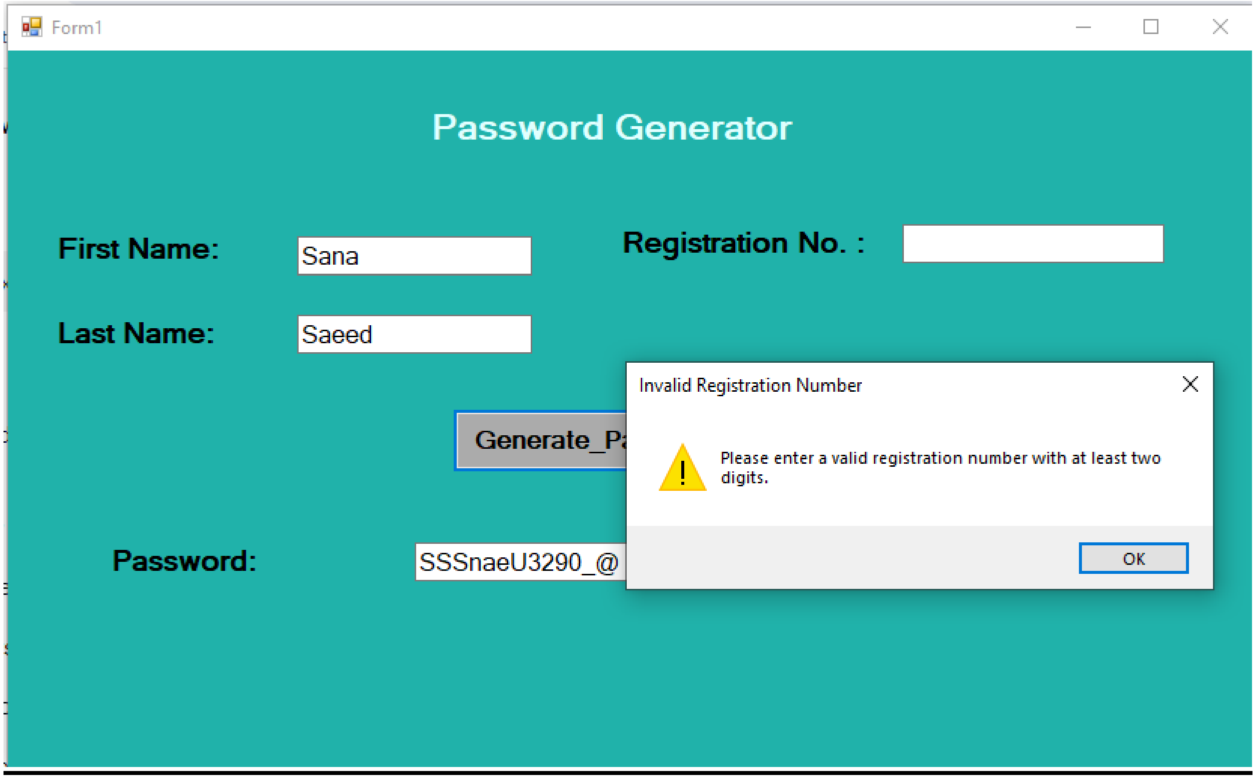
Last Name:

Registration No. : sp21-bcs-032

Password: SSSnaeU3290_@

Error

Please enter your first and last name.



Password Generator

First Name: Sana

Last Name: Saeed

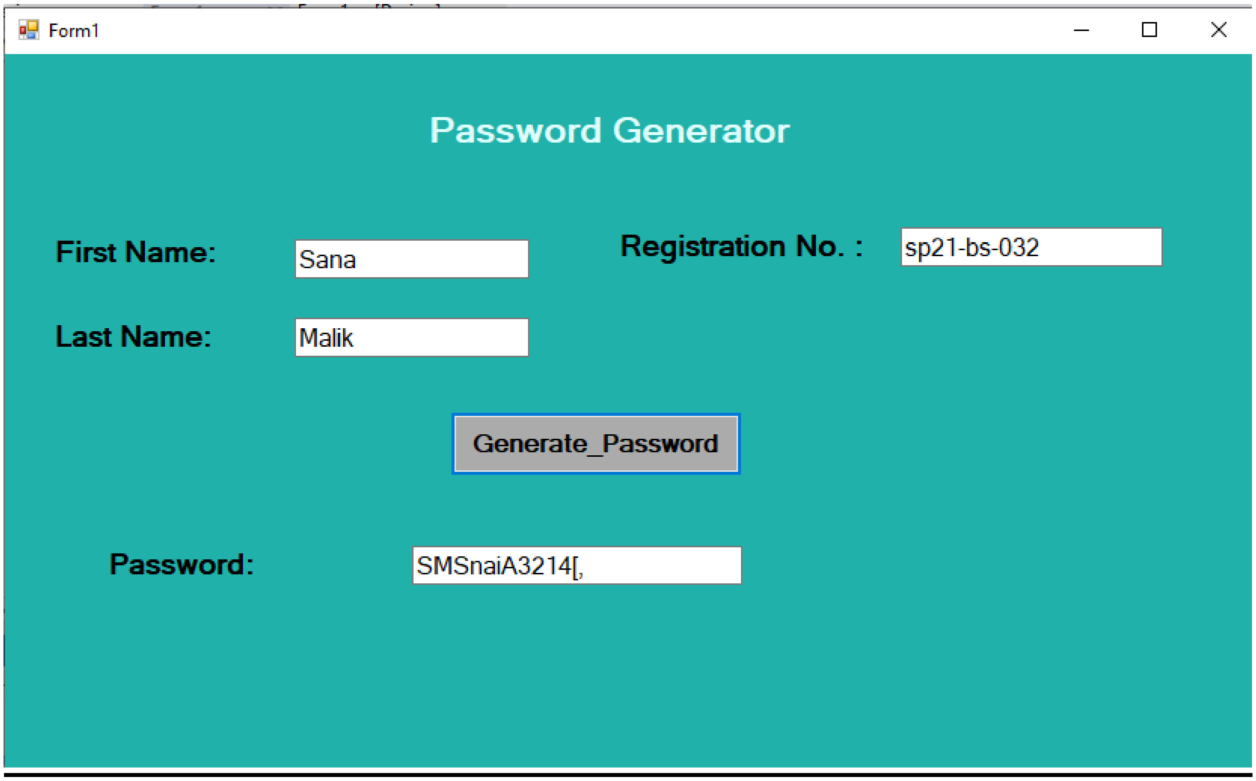
Registration No. :

Password: SSSnaeU3290_@

Invalid Registration Number

Please enter a valid registration number with at least two digits.

Generated Passwords:



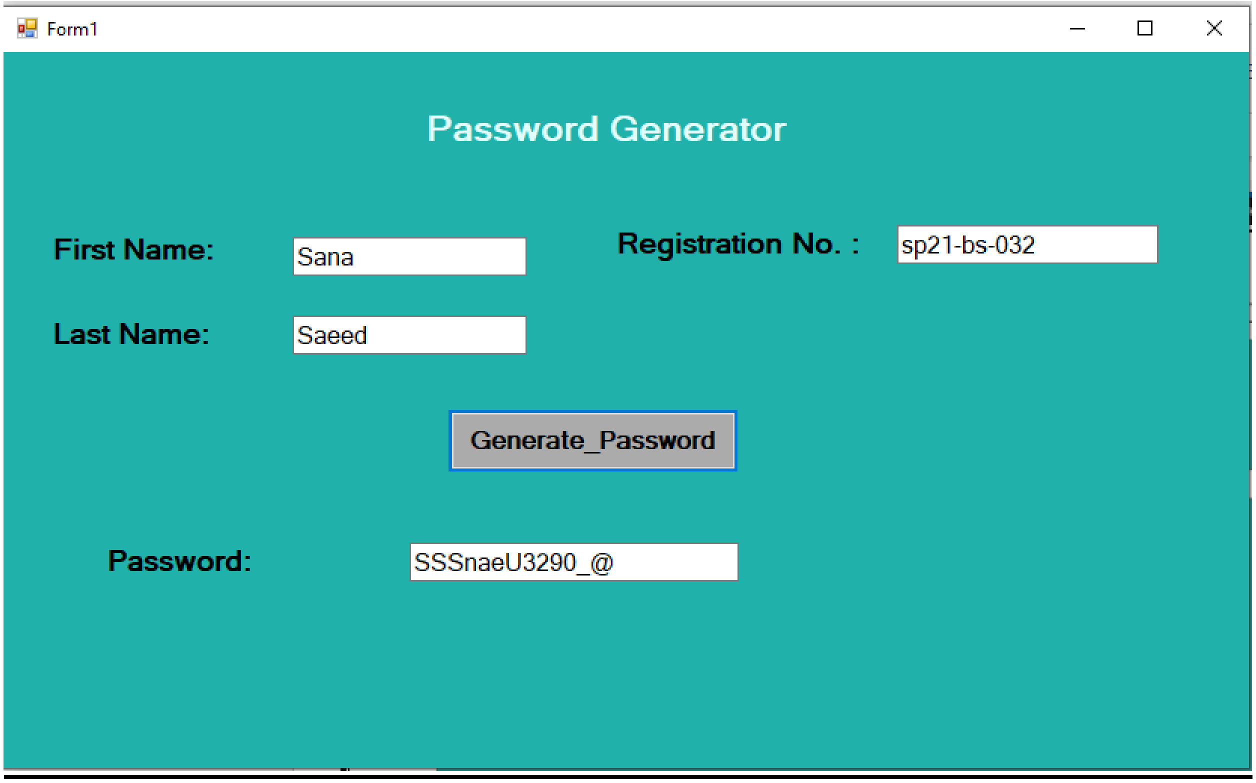
Password Generator

First Name: Sana

Last Name: Malik

Registration No. : sp21-bs-032

Password: SMSnaiA3214[,



Password Generator

First Name: Sana

Registration No. : sp21-bs-032

Last Name: Saeed

Password: SSSnaeU3290_@