Comsats University Islamabad

Attock Campus

Lab Task 1, 2 , 3

Name:

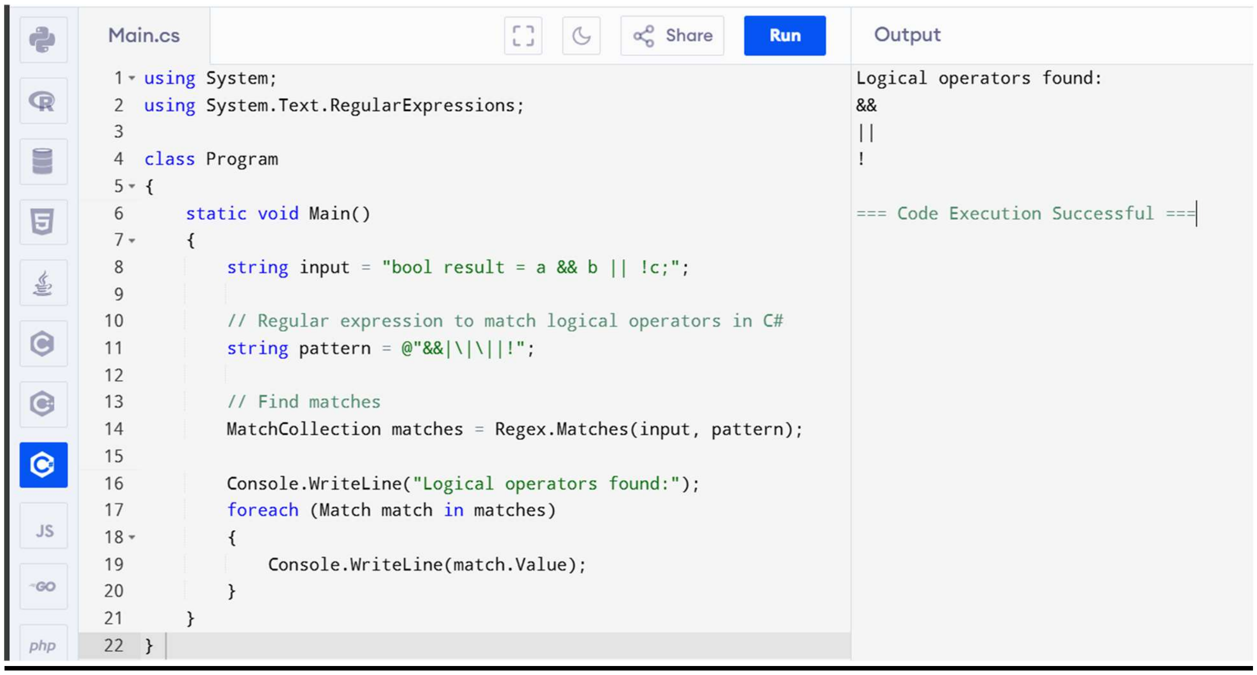
Muhammad Anas (Sp22-Bcs-042)

Submitted To:

Sir Bilah Haider

Date:7th March,2025

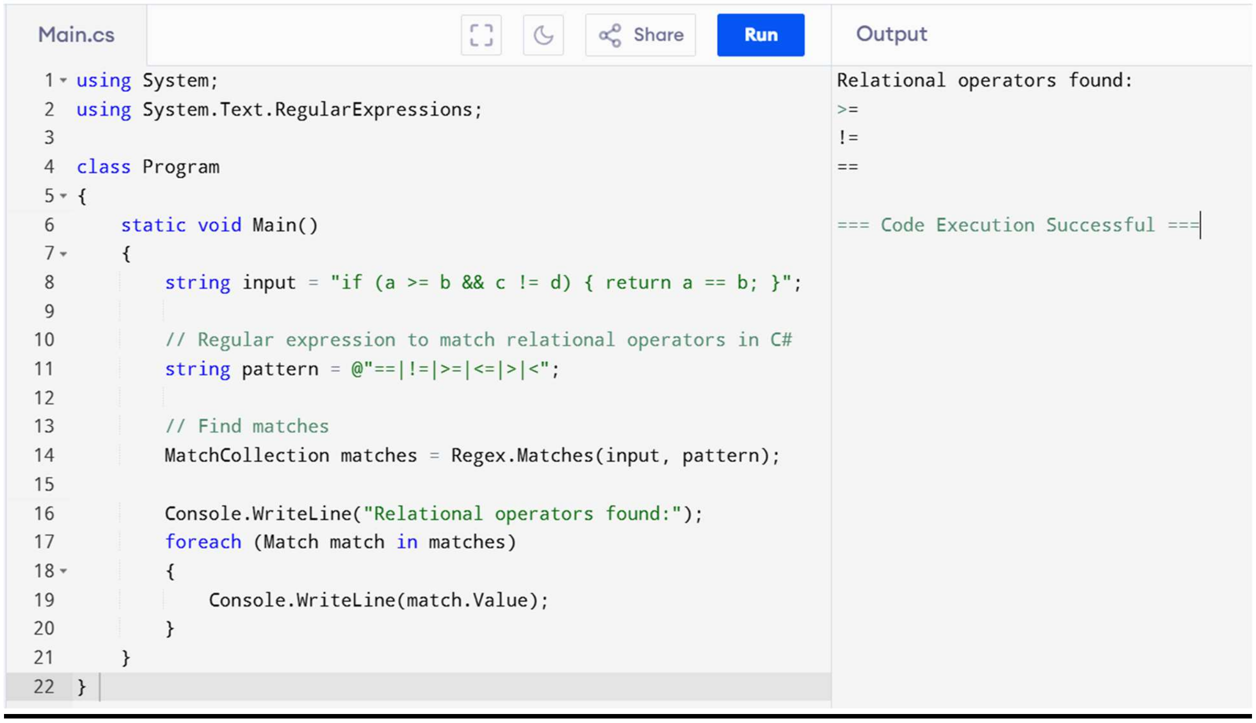
Lab 1 Task 1:



```csharp
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string input = "bool result = a && b || !c;";

        // Regular expression to match logical operators in C#
        string pattern = @"&&|\|\||!";

        // Find matches
        MatchCollection matches = Regex.Matches(input, pattern);

        Console.WriteLine("Logical operators found:");
        foreach (Match match in matches)
        {
            Console.WriteLine(match.Value);
        }
    }
}
```

```
Logical operators found:
&&
||
!

=== Code Execution Successful ===
```

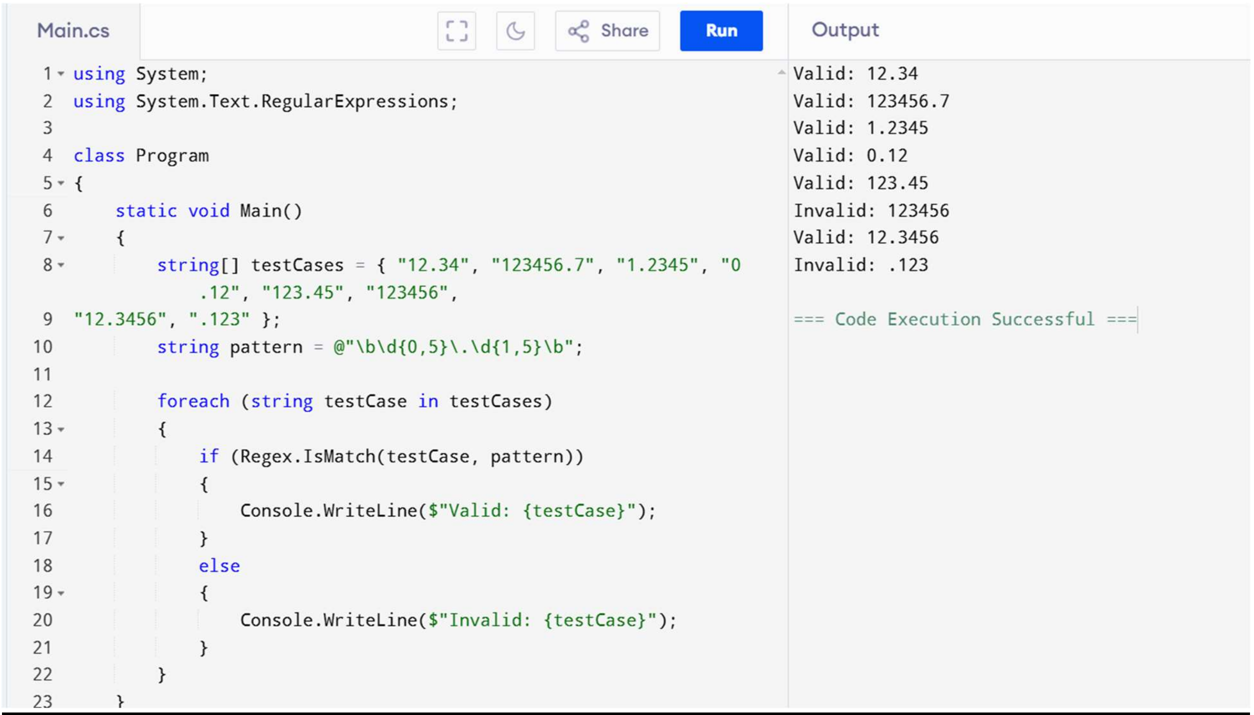
Task 2:



```csharp
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string input = "if (a >= b && c != d) { return a == b; }";

        // Regular expression to match relational operators in C#
        string pattern = @"==|!=|>=|<=|>|<";

        // Find matches
        MatchCollection matches = Regex.Matches(input, pattern);

        Console.WriteLine("Relational operators found:");
        foreach (Match match in matches)
        {
            Console.WriteLine(match.Value);
        }
    }
}
```

```
Relational operators found:
>=
!=
==

=== Code Execution Successful ===
```

Task 3:



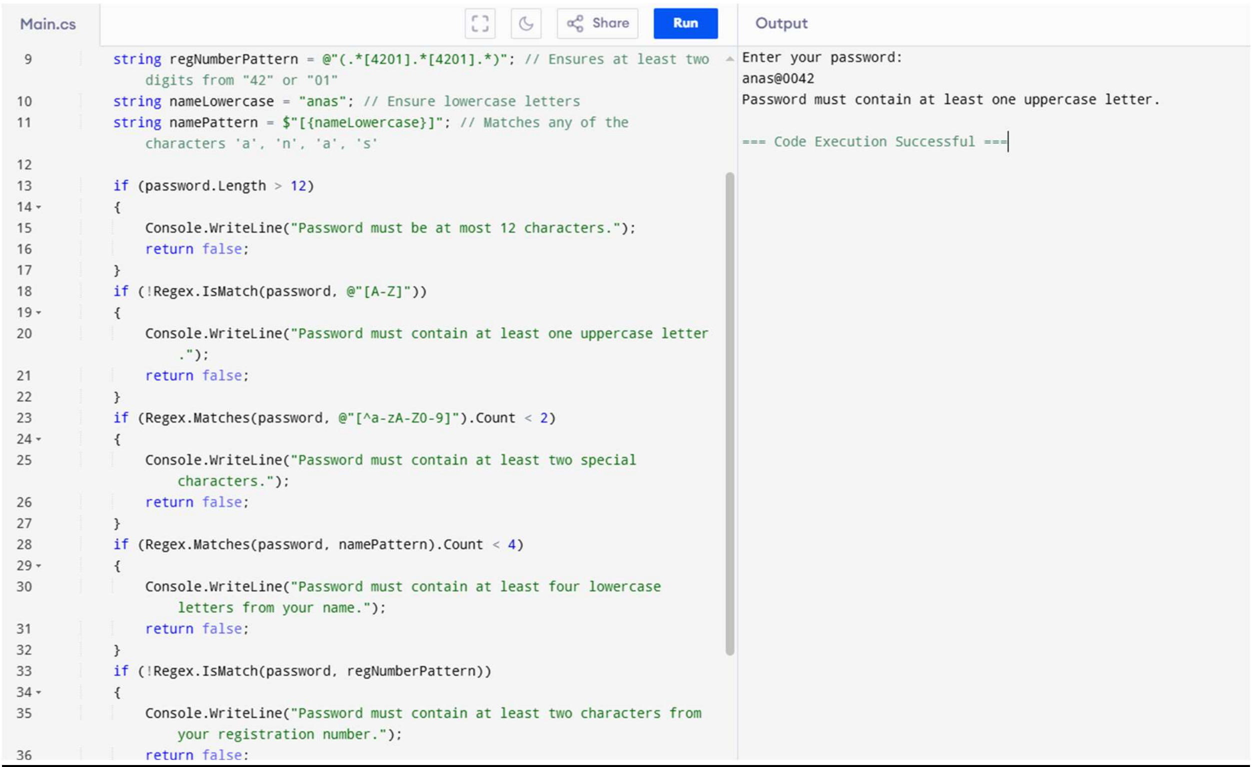
Main.cs

```
using System;
using System.Text.RegularExpressions;

class Program
{
    static void Main()
    {
        string[] testCases = { "12.34", "123456.7", "1.2345", "0
            .12", "123.45", "123456",
"12.3456", ".123" };
        string pattern = @"\b\d{0,5}\.\d{1,5}\b";

        foreach (string testCase in testCases)
        {
            if (Regex.IsMatch(testCase, pattern))
            {
                Console.WriteLine($"Valid: {testCase}");
            }
            else
            {
                Console.WriteLine($"Invalid: {testCase}");
            }
        }
    }
```

Output

```
Valid: 12.34
Valid: 123456.7
Valid: 1.2345
Valid: 0.12
Valid: 123.45
Invalid: 123456
Valid: 12.3456
Invalid: .123

=== Code Execution Successful ===
```

Lab 2 Task 1:



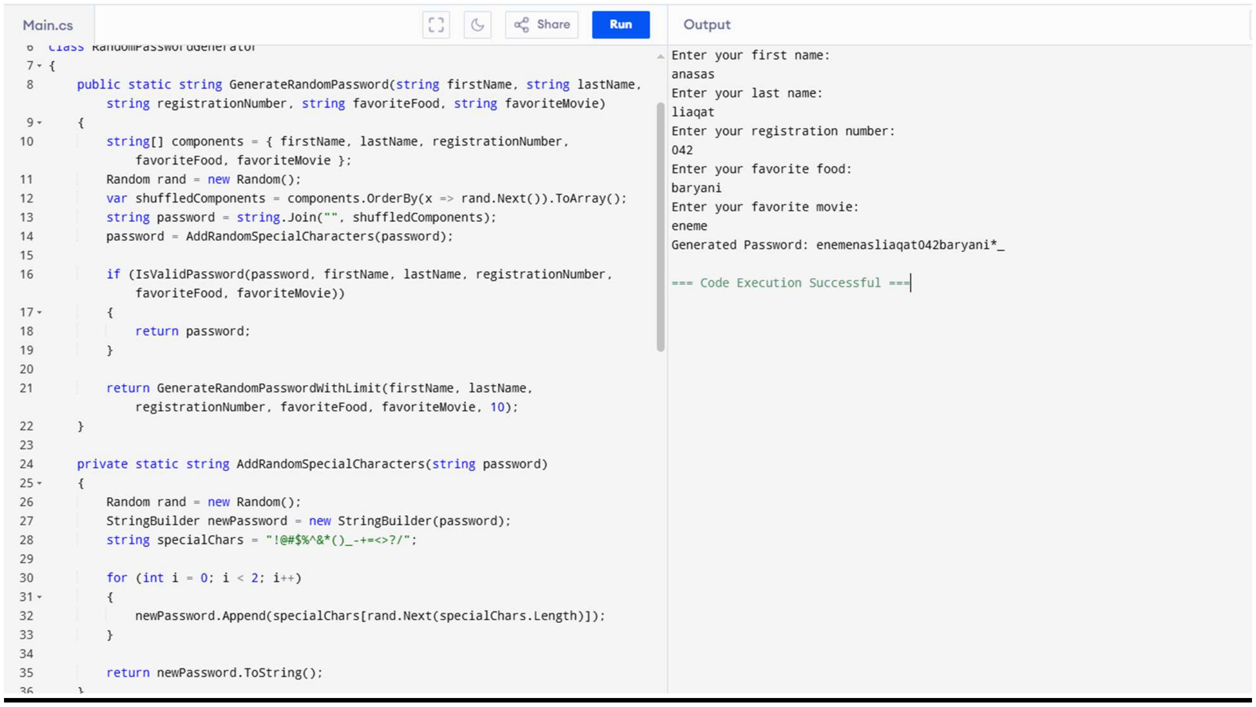
Main.cs

```
    string regNumberPattern = @"(.*[4201].*[4201].*)"; // Ensures at least two
        digits from "42" or "01"
    string nameLowercase = "anas"; // Ensure lowercase letters
    string namePattern = $"[{nameLowercase}]"; // Matches any of the
        characters 'a', 'n', 'a', 's'

    if (password.Length > 12)
    {
        Console.WriteLine("Password must be at most 12 characters.");
        return false;
    }
    if (!Regex.IsMatch(password, @"[A-Z]"))
    {
        Console.WriteLine("Password must contain at least one uppercase letter
            .");
        return false;
    }
    if (Regex.Matches(password, @"[^a-zA-Z0-9]").Count < 2)
    {
        Console.WriteLine("Password must contain at least two special
            characters.");
        return false;
    }
    if (Regex.Matches(password, namePattern).Count < 4)
    {
        Console.WriteLine("Password must contain at least four lowercase
            letters from your name.");
        return false;
    }
    if (!Regex.IsMatch(password, regNumberPattern))
    {
        Console.WriteLine("Password must contain at least two characters from
            your registration number.");
        return false;
```

Output

```
Enter your password:
anas@0042
Password must contain at least one uppercase letter.

=== Code Execution Successful ===
```

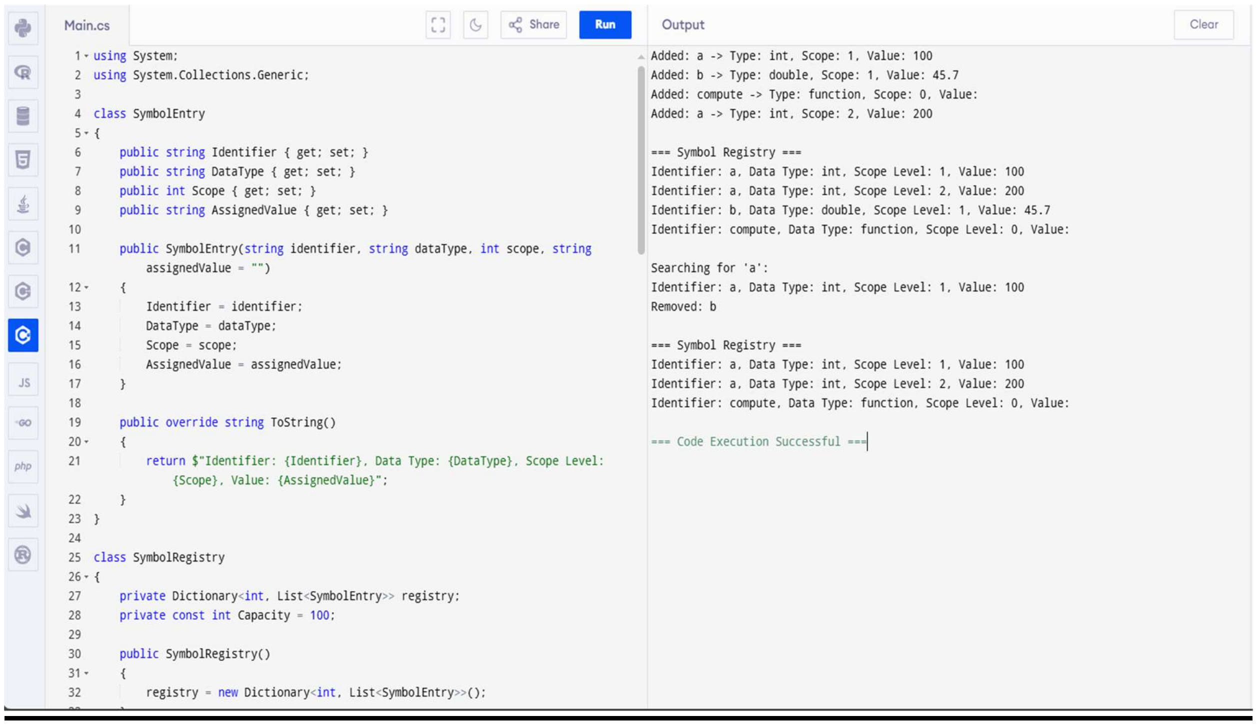
Task 2:



```csharp
class RandomPasswordGenerator
{
    public static string GenerateRandomPassword(string firstName, string lastName,
        string registrationNumber, string favoriteFood, string favoriteMovie)
    {
        string[] components = { firstName, lastName, registrationNumber,
            favoriteFood, favoriteMovie };
        Random rand = new Random();
        var shuffledComponents = components.OrderBy(x => rand.Next()).ToArray();
        string password = string.Join("", shuffledComponents);
        password = AddRandomSpecialCharacters(password);

        if (IsValidPassword(password, firstName, lastName, registrationNumber,
            favoriteFood, favoriteMovie))
        {
            return password;
        }

        return GenerateRandomPasswordWithLimit(firstName, lastName,
            registrationNumber, favoriteFood, favoriteMovie, 10);
    }

    private static string AddRandomSpecialCharacters(string password)
    {
        Random rand = new Random();
        StringBuilder newPassword = new StringBuilder(password);
        string specialChars = "!@#$%^&*()_-+=<>?/";

        for (int i = 0; i < 2; i++)
        {
            newPassword.Append(specialChars[rand.Next(specialChars.Length)]);
        }

        return newPassword.ToString();
    }
```

```
Enter your first name:
anasas
Enter your last name:
liaqat
Enter your registration number:
042
Enter your favorite food:
baryani
Enter your favorite movie:
eneme
Generated Password: enemenasliaqat042baryani*_

=== Code Execution Successful ===
```

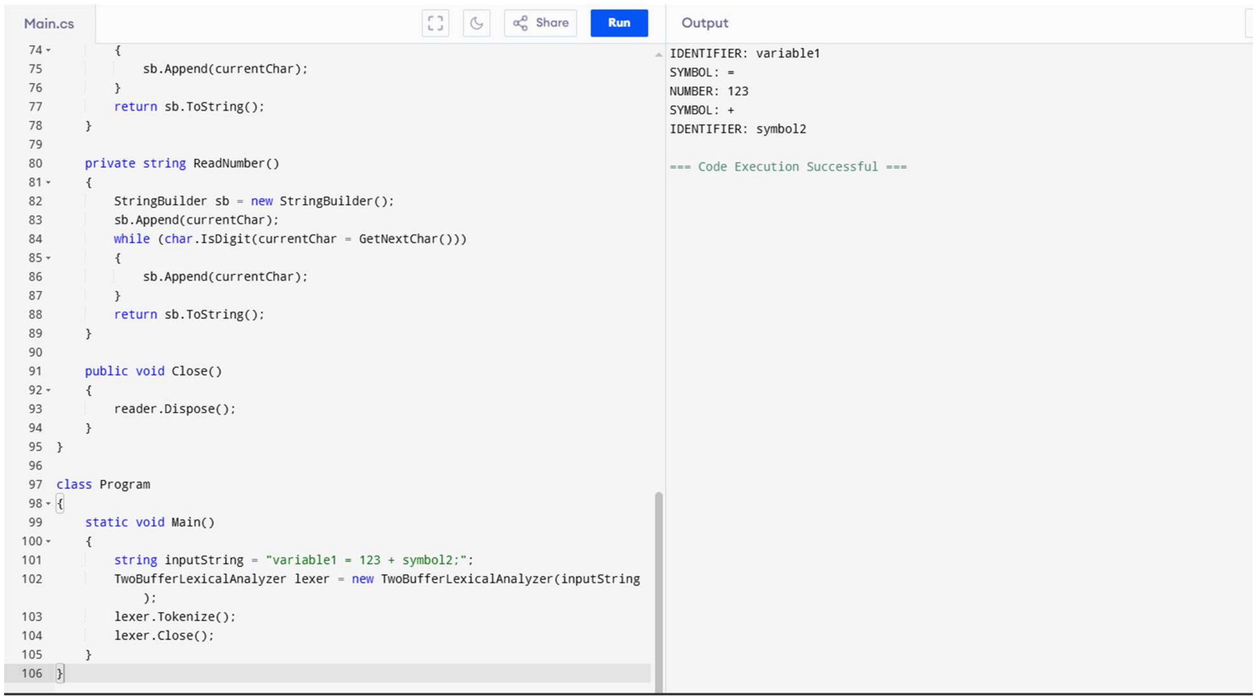
Lab 3 Task 1;



```csharp
using System;
using System.Collections.Generic;

class SymbolEntry
{
    public string Identifier { get; set; }
    public string DataType { get; set; }
    public int Scope { get; set; }
    public string AssignedValue { get; set; }

    public SymbolEntry(string identifier, string dataType, int scope, string
        assignedValue = "")
    {
        Identifier = identifier;
        DataType = dataType;
        Scope = scope;
        AssignedValue = assignedValue;
    }

    public override string ToString()
    {
        return $"Identifier: {Identifier}, Data Type: {DataType}, Scope Level:
            {Scope}, Value: {AssignedValue}";
    }
}

class SymbolRegistry
{
    private Dictionary<int, List<SymbolEntry>> registry;
    private const int Capacity = 100;

    public SymbolRegistry()
    {
        registry = new Dictionary<int, List<SymbolEntry>>();
```

```
Added: a -> Type: int, Scope: 1, Value: 100
Added: b -> Type: double, Scope: 1, Value: 45.7
Added: compute -> Type: function, Scope: 0, Value:
Added: a -> Type: int, Scope: 2, Value: 200

=== Symbol Registry ===
Identifier: a, Data Type: int, Scope Level: 1, Value: 100
Identifier: a, Data Type: int, Scope Level: 2, Value: 200
Identifier: b, Data Type: double, Scope Level: 1, Value: 45.7
Identifier: compute, Data Type: function, Scope Level: 0, Value:

Searching for 'a':
Identifier: a, Data Type: int, Scope Level: 1, Value: 100
Removed: b

=== Symbol Registry ===
Identifier: a, Data Type: int, Scope Level: 1, Value: 100
Identifier: a, Data Type: int, Scope Level: 2, Value: 200
Identifier: compute, Data Type: function, Scope Level: 0, Value:

=== Code Execution Successful ===
```

Task 2:



```
            {
                sb.Append(currentChar);
            }
            return sb.ToString();
        }

        private string ReadNumber()
        {
            StringBuilder sb = new StringBuilder();
            sb.Append(currentChar);
            while (char.IsDigit(currentChar = GetNextChar()))
            {
                sb.Append(currentChar);
            }
            return sb.ToString();
        }

        public void Close()
        {
            reader.Dispose();
        }
    }

    class Program
    {
        static void Main()
        {
            string inputString = "variable1 = 123 + symbol2;";
            TwoBufferLexicalAnalyzer lexer = new TwoBufferLexicalAnalyzer(inputString
                );
            lexer.Tokenize();
            lexer.Close();
        }
    }
```

```
IDENTIFIER: variable1
SYMBOL: =
NUMBER: 123
SYMBOL: +
IDENTIFIER: symbol2

=== Code Execution Successful ===
```