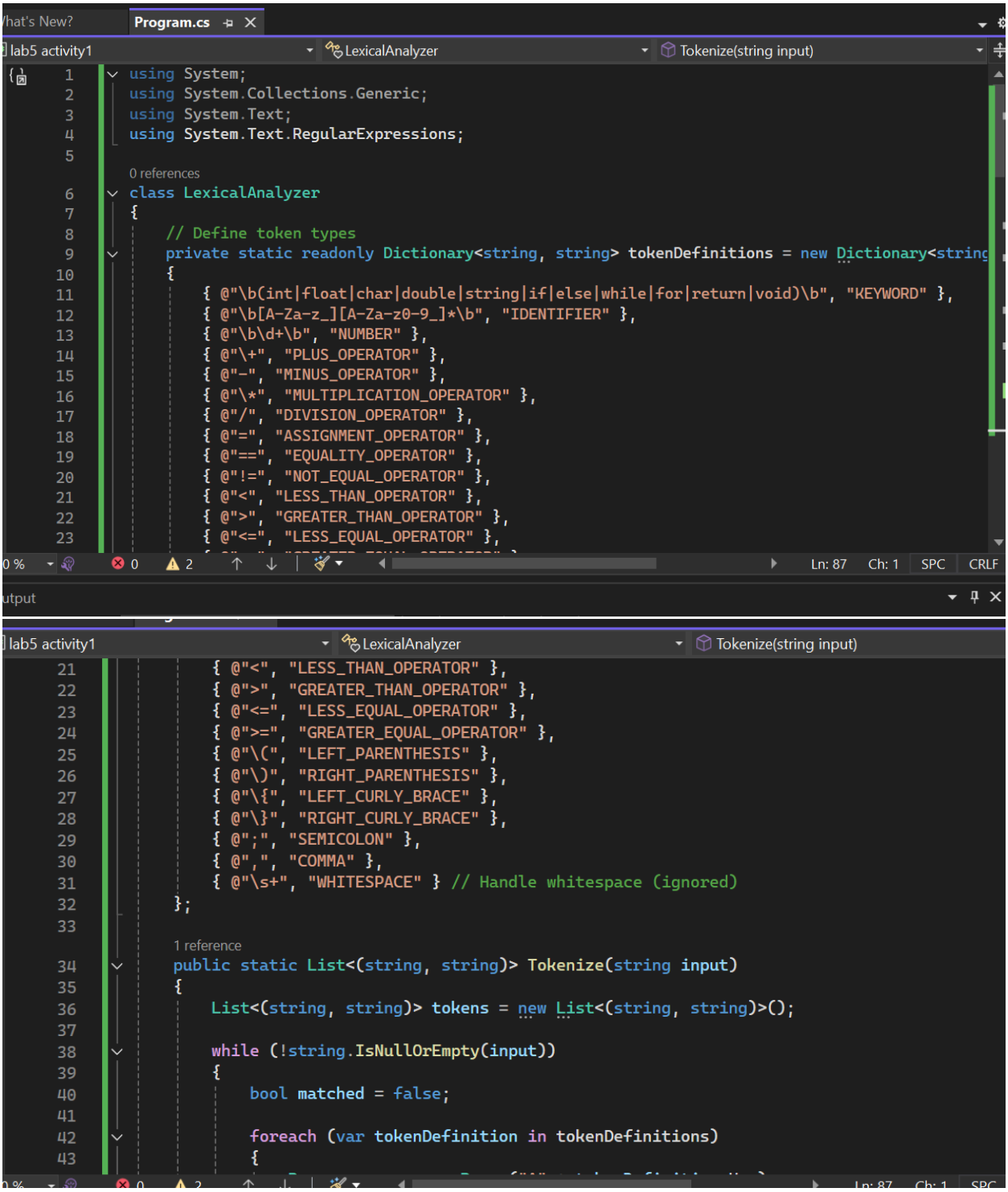
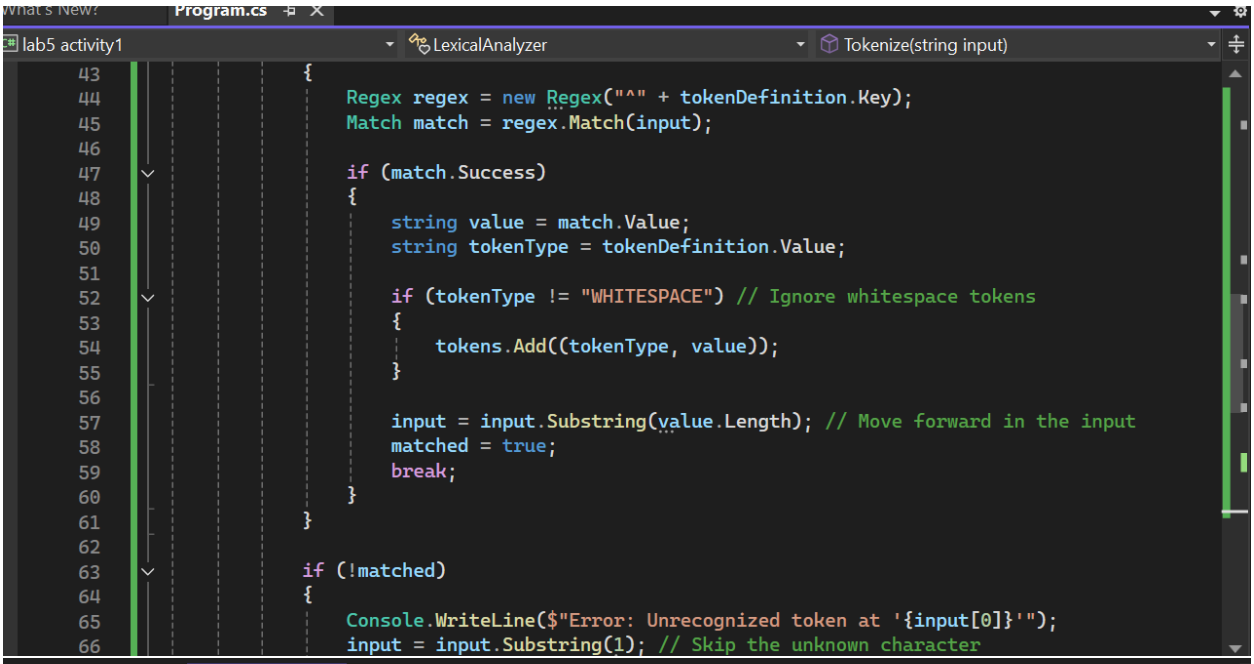
LAB NO 5:

Activity 1:

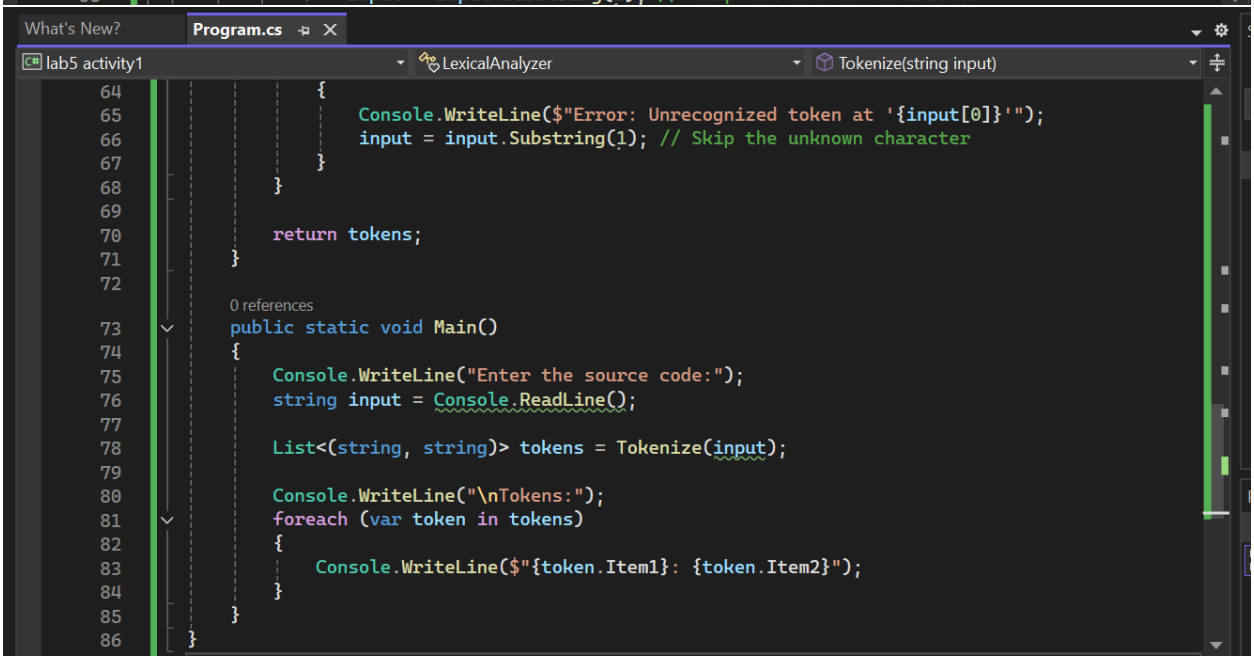


```
using System;
using System.Collections.Generic;
using System.Text;
using System.Text.RegularExpressions;

0 references
class LexicalAnalyzer
{
    // Define token types
    private static readonly Dictionary<string, string> tokenDefinitions = new Dictionary<string
    {
        { @"\b(int|float|char|double|string|if|else|while|for|return|void)\b", "KEYWORD" },
        { @"\b[A-Za-z_][A-Za-z0-9_]*\b", "IDENTIFIER" },
        { @"\b\d+\b", "NUMBER" },
        { @"\+", "PLUS_OPERATOR" },
        { @"-", "MINUS_OPERATOR" },
        { @"\*", "MULTIPLICATION_OPERATOR" },
        { @"/", "DIVISION_OPERATOR" },
        { @"=", "ASSIGNMENT_OPERATOR" },
        { @"==", "EQUALITY_OPERATOR" },
        { @"!=", "NOT_EQUAL_OPERATOR" },
        { @"<", "LESS_THAN_OPERATOR" },
        { @">", "GREATER_THAN_OPERATOR" },
        { @"<=", "LESS_EQUAL_OPERATOR" },
```

```
        { @"<", "LESS_THAN_OPERATOR" },
        { @">", "GREATER_THAN_OPERATOR" },
        { @"<=", "LESS_EQUAL_OPERATOR" },
        { @">=", "GREATER_EQUAL_OPERATOR" },
        { @"\(", "LEFT_PARENTHESIS" },
        { @"\)", "RIGHT_PARENTHESIS" },
        { @"\{", "LEFT_CURLY_BRACE" },
        { @"\}", "RIGHT_CURLY_BRACE" },
        { @";", "SEMICOLON" },
        { @",", "COMMA" },
        { @"\s+", "WHITESPACE" } // Handle whitespace (ignored)
    };

    1 reference
    public static List<(string, string)> Tokenize(string input)
    {
        List<(string, string)> tokens = new List<(string, string)>();

        while (!string.IsNullOrEmpty(input))
        {
            bool matched = false;

            foreach (var tokenDefinition in tokenDefinitions)
            {
```



```
                {
                    Regex regex = new Regex("^" + tokenDefinition.Key);
                    Match match = regex.Match(input);

                    if (match.Success)
                    {
                        string value = match.Value;
                        string tokenType = tokenDefinition.Value;

                        if (tokenType != "WHITESPACE") // Ignore whitespace tokens
                        {
                            tokens.Add((tokenType, value));
                        }

                        input = input.Substring(value.Length); // Move forward in the input
                        matched = true;
                        break;
                    }
                }

                if (!matched)
                {
                    Console.WriteLine($"Error: Unrecognized token at '{input[0]}'");
                    input = input.Substring(1); // Skip the unknown character
```

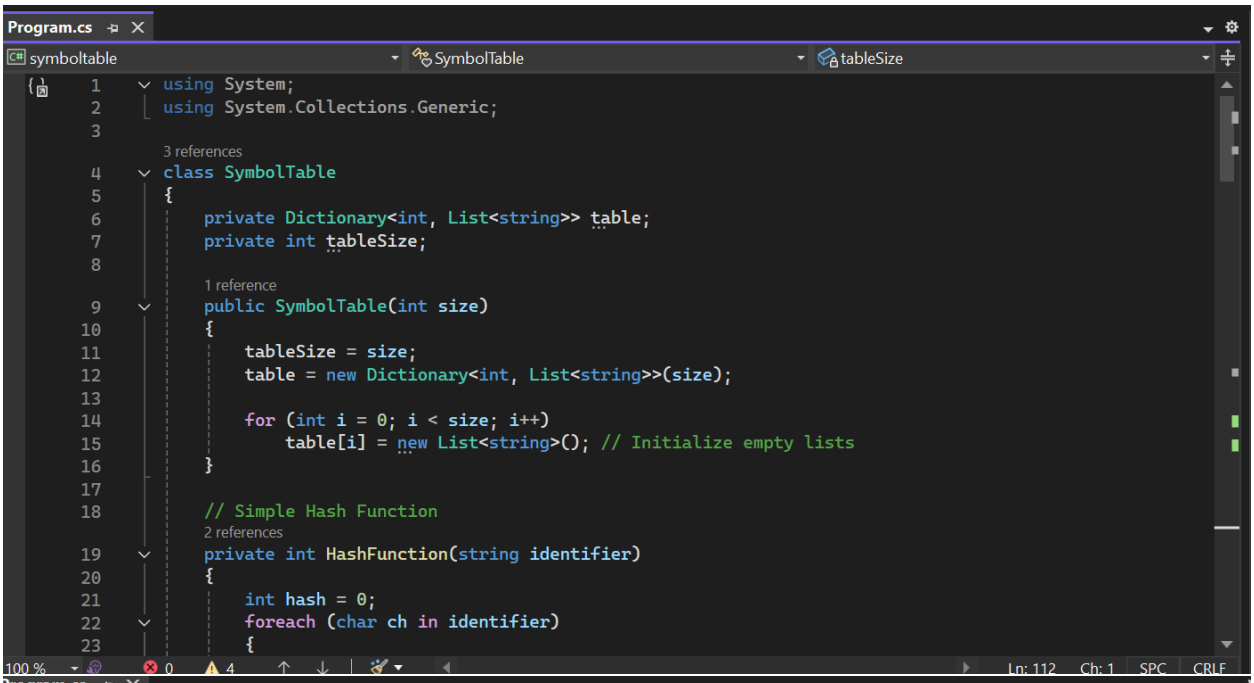


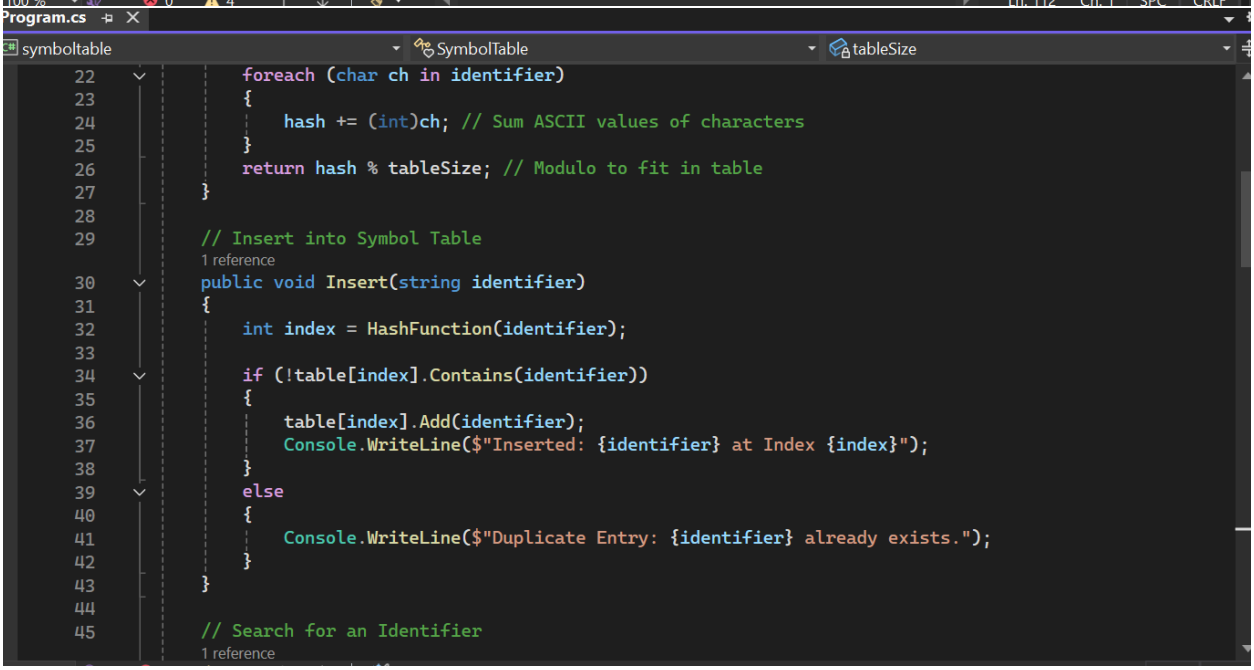
```
                {
                    Console.WriteLine($"Error: Unrecognized token at '{input[0]}'");
                    input = input.Substring(1); // Skip the unknown character
                }
            }

            return tokens;
        }

        0 references
        public static void Main()
        {
            Console.WriteLine("Enter the source code:");
            string input = Console.ReadLine();

            List<(string, string)> tokens = Tokenize(input);

            Console.WriteLine("\nTokens:");
            foreach (var token in tokens)
            {
                Console.WriteLine($"{token.Item1}: {token.Item2}");
            }
        }
    }
```

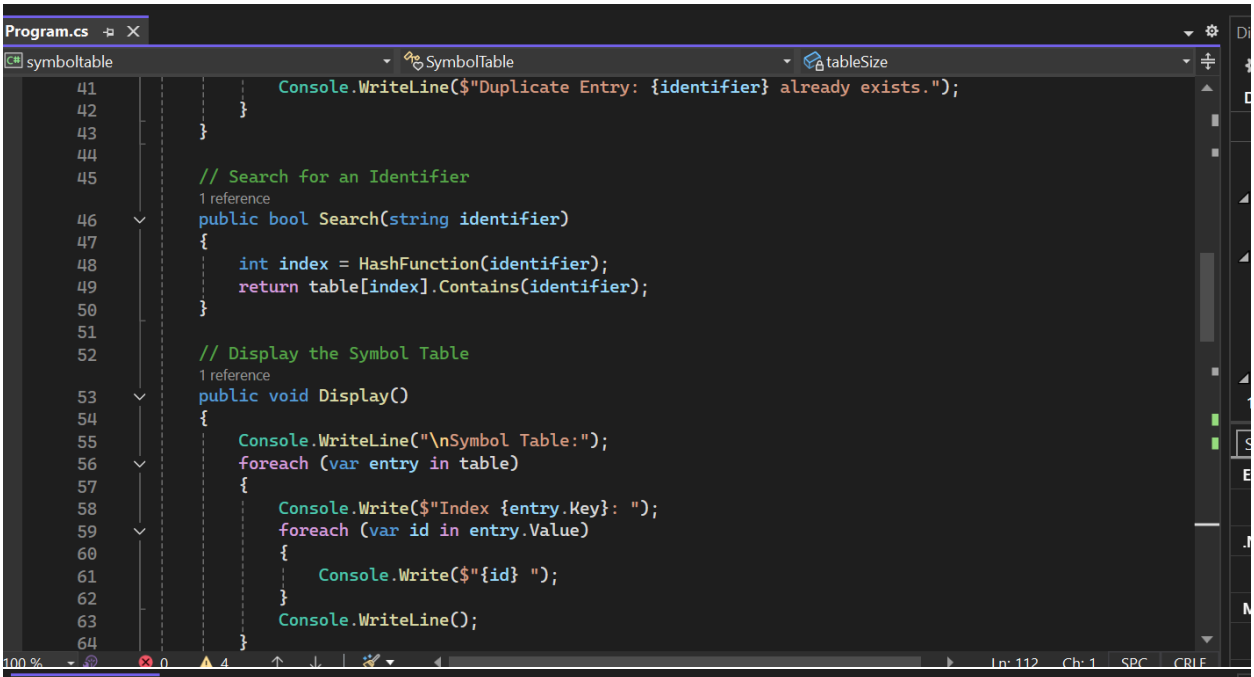
Output:

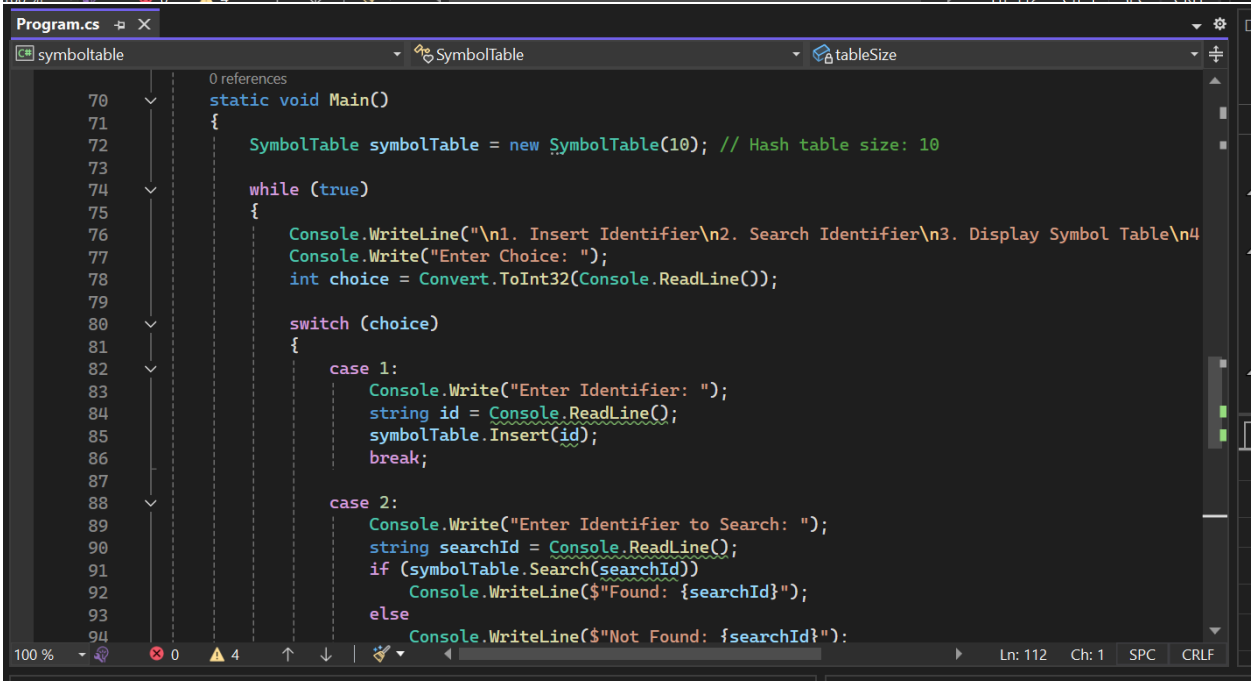
```
Enter the source code:
 int x = 10;

Tokens:
KEYWORD: int
IDENTIFIER: x
ASSIGNMENT_OPERATOR: =
NUMBER: 10
SEMICOLON: ;

C:\Users\HP\source\repos\lab5 activity1\lab5 activity1\bin\Debug\net8.0\lab5 activity1.exe (process 15928) exited with code 0 (0x
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when d
ging stops.
Press any key to close this window . . .
```

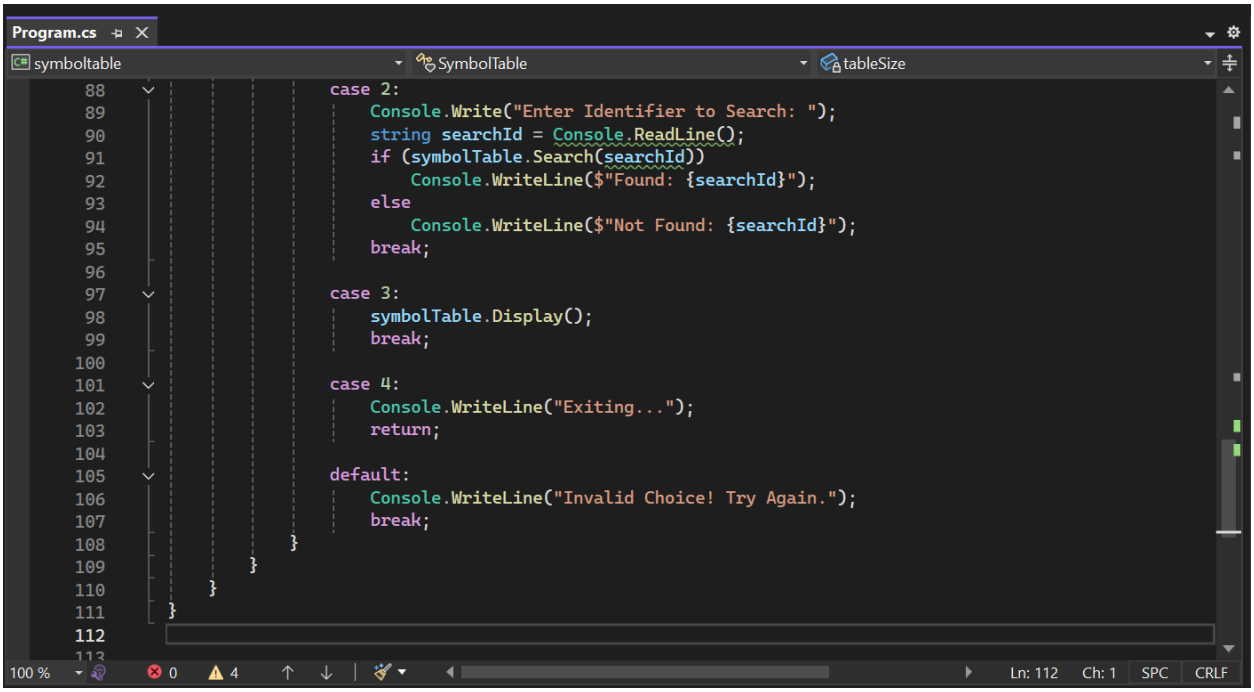
Graded Task 1:






Program.cs
symboltable
SymbolTable
tableSize
Console.WriteLine($"Duplicate Entry: {identifier} already exists.");
}
}
// Search for an Identifier
1 reference
public bool Search(string identifier)
{
int index = HashFunction(identifier);
return table[index].Contains(identifier);
}
// Display the Symbol Table
1 reference
public void Display()
{
Console.WriteLine("\nSymbol Table:");
foreach (var entry in table)
{
Console.Write($"Index {entry.Key}: ");
foreach (var id in entry.Value)
{
Console.Write($"{id} ");
}
Console.WriteLine();
}
100 %
0
4
Ln: 112
Ch: 1
SPC
CRLF


Program.cs
symboltable
SymbolTable
tableSize
0 references
static void Main()
{
SymbolTable symbolTable = new SymbolTable(10); // Hash table size: 10
while (true)
{
Console.WriteLine("\n1. Insert Identifier\n2. Search Identifier\n3. Display Symbol Table\n4
Console.Write("Enter Choice: ");
int choice = Convert.ToInt32(Console.ReadLine());
switch (choice)
{
case 1:
Console.Write("Enter Identifier: ");
string id = Console.ReadLine();
symbolTable.Insert(id);
break;
case 2:
Console.Write("Enter Identifier to Search: ");
string searchId = Console.ReadLine();
if (symbolTable.Search(searchId))
Console.WriteLine($"Found: {searchId}");
else
Console.WriteLine($"Not Found: {searchId}");
100 %
0
4
Ln: 112
Ch: 1
SPC
CRLF


Program.cs
symboltable
SymbolTable
tableSize
case 2:
Console.Write("Enter Identifier to Search: ");
string searchId = Console.ReadLine();
if (symbolTable.Search(searchId))
Console.WriteLine($"Found: {searchId}");
else
Console.WriteLine($"Not Found: {searchId}");
break;
case 3:
symbolTable.Display();
break;
case 4:
Console.WriteLine("Exiting...");
return;
default:
Console.WriteLine("Invalid Choice! Try Again.");
break;
100 %
0
4
Ln: 112
Ch: 1
SPC
CRLF

Output:

```
1. Insert Identifier
2. Search Identifier
3. Display Symbol Table
4. Exit
Enter Choice: 1
Enter Identifier: x
Inserted: x at Index 0

1. Insert Identifier
2. Search Identifier
3. Display Symbol Table
4. Exit
Enter Choice: 1
Enter Identifier: y
Inserted: y at Index 1

1. Insert Identifier
2. Search Identifier
3. Display Symbol Table
4. Exit
Enter Choice: 3

Symbol Table:
Index 0: x
Index 1: y
Index 2:
Index 3:
Index 4:
Index 5:
Index 6:
Index 7:
Index 8:
```

```
Index 7:
Index 8:
Index 9:

1. Insert Identifier
2. Search Identifier
3. Display Symbol Table
4. Exit
Enter Choice: 4
Exiting...

C:\Users\HP\source\repos\symboltable\symboltable\bin\Debug\net8.0\symboltable.exe (process 18772) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debug
ging stops.
Press any key to close this window . . .
```