Language Map for C#

Variable Declaration
Is this language strongly typed or dynamically typed? Provide at least three examples (with different data types or keywords) of how variables are declared in this language.
	C# is strongly typed.
	Examples of declared variables:
	int Age = 40;
	string myName = “Adam”;
	double wingSpan = 12.4;

Data Types *List all of the data types (and ranges) supported by this language.*	int: whole numbers from -2,147,483,648 to 2,147,483,647 long: whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 float: fractions with 7 decimal digits double: fractions with 15 decimal digits bool: Boolean true/false char: single character/letter, surrounded by single quotes string: sequence of characters, surrounded by double quotes

Selection Structures
Provide examples of all selection structures supported by this language (if, if else, etc.) ***Don’t just list them, show code samples of how each would look in a real program.***
	If statement: allows you to execute a block of code if a condition is true.
	`if (something is true)`
	`{`
	`  // do this`
	`}`
	
	If-else statement: allows you to execute one block of code if a condition is true and another if it's false.
	`if(value == 10)`
	`{`
	`//do this`
	`}`
	`else`
	`{`
	`//do this`
	`}`
	
	If-else if-else statement: handles multiple conditions
	`int score = 75;`
	`if (score >= 90)`
	`{`
	`   Console.WriteLine("You got an A.");`
	`}`
	`else if (score >= 80)`
	`{`

```
    Console.WriteLine("You got a B.");
}
else if (score >= 70)
{
    Console.WriteLine("You got a C.");
}
else
{
    Console.WriteLine("You need to study harder.");
}
```

Switch statement: used to evaluate a variable against a list of values and execute code based on the match.

```
int day = 3;
switch (day)
{
    case 1:
        Console.WriteLine("Monday");
        break;
    case 2:
        Console.WriteLine("Tuesday");
        break;
    case 3:
        Console.WriteLine("Wednesday");
        break;
    default:
        Console.WriteLine("Other day");
        break;
}
```

Repetition Structures

Provide examples of all repetition structures supported by this language (loops, etc.) ***Don't just list them, show code samples of how each would look in a real program.***

For Loop: used for a definite number of iterations.

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine("Iteration " + i);
}
```

While loop: repeats a block of code as long as a condition is true.

```
int count = 0;
while (count < 3)
{
    Console.WriteLine("Count: " + count);
```

	`    count++;`
	`}`
	
	Do-while loop: similar to the while loop, but it ensures that the block of code is executed at least once.
	`int number;`
	`do`
	`{`
	`    Console.Write("Enter a positive number: ");`
	`} while (!int.TryParse(Console.ReadLine(), out number) || number <= 0);`
	`Console.WriteLine("You entered a positive number: " + number);`
	
	Foreach loop: used to iterate over elements in a collection or array.
	`int[] numbers = { 1, 2, 3, 4, 5 };`
	`foreach (int num in numbers)`
	`{`
	`    Console.WriteLine("Number: " + num);`
	`}`
	
	Break statement: used to exit a loop prematurely when a certain condition is met.
	`for (int i = 0; i < 10; i++)`
	`{`
	`    if (i == 5)`
	`    {`
	`        Console.WriteLine("Breaking the loop at i = 5");`
	`        break;`
	`    }`
	`    Console.WriteLine("Iteration " + i);`
	`}`
	
	Continue statement: used to skip the current iteration of a loop and continue to the next one.
	`for (int i = 0; i < 5; i++)`
	`{`
	`    if (i == 2)`
	`    {`
	`        Console.WriteLine("Skipping iteration at i = 2");`
	`        continue;`
	`    }`
	`    Console.WriteLine("Iteration " + i);`
	`}`

Arrays	Yes, C# supports arrays.

If this language supports arrays, provide ***at least two examples*** *of creating an array with a primitive or String data types (e.g. float, int, String, etc.) If the language supports declaring arrays in multiple ways, provide an example of way.*	Array initializer:
	`int[] intArray = { 1, 2, 3, 4, 5 };`
	
	`string[] stringArray = { "apple", "banana", "cherry" };`
	
	Using the 'new' keyword:
	`int[] intArray = new int[5]; // Creates an integer array with 5 elements`
	
	`string[] stringArray = new string[3]; // Creates a string array with 3 elements`
	
	Initializing arrays with values after creation:
	`int[] intArray = new int[3]; // Creates an integer array with 3 elements`
	`intArray[0] = 10;`
	`intArray[1] = 20;`
	`intArray[2] = 30;`
	
	multidimensional arrays:
	`int[,] twoDArray = new int[2, 3];`
	`twoDArray[0, 0] = 1;`
	`twoDArray[0, 1] = 2;`
	`twoDArray[0, 2] = 3;`
	`twoDArray[1, 0] = 4;`
	`twoDArray[1, 1] = 5;`
	`twoDArray[1, 2] = 6;`

Data Structures *If this language provides a standard set of data structures, provide a list of the data structures and their Big-Oh complexity (identify what the complexity represents).*	1. **Arrays**: • Access: O(1) • Search (unsorted): O(n) • Search (sorted with binary search): O(log n) • Insert/Delete (at the beginning): O(n) • Insert/Delete (in the middle or end): O(n) 2. **Lists (List<T>)**: • Access: O(1) • Search: O(n) • Insert/Delete (at the beginning or middle): O(n) • Insert/Delete (at the end): Amortized O(1) 3. **Linked Lists (LinkedList<T>)**: • Access: O(n) • Search: O(n)

	• Insert/Delete (any position): O(1) with a reference 4. **Stacks (Stack<T>)**: • Push (Insert): O(1) • Pop (Delete): O(1) • Peek (Access top): O(1) 5. **Queues (Queue<T>)**: • Enqueue (Insert): O(1) • Dequeue (Delete): O(1) • Peek (Access front): O(1) 6. **Dictionaries (Dictionary<TKey, TValue>)**: • Search (by key): O(1) on average (O(n) in worst-case for hash collisions) • Insert/Delete (by key): O(1) on average (O(n) in worst-case for hash collisions) 7. **Sorted Dictionaries (SortedDictionary<TKey, TValue>)**: • Search (by key): O(log n) • Insert/Delete (by key): O(log n) 8. **Sets (HashSet<T>)**: • Search: O(1) on average (O(n) in worst-case for hash collisions) • Insert/Delete: O(1) on average (O(n) in worst-case for hash collisions) 9. **Sorted Sets (SortedSet<T>)**: • Search: O(log n) • Insert/Delete: O(log n) 10. **Stacks and Queues (using LinkedList or Array)**: • The time complexities depend on the underlying data structure used. 11. **Bit Arrays (BitArray)**: • Access/Update individual bits: O(1) 12. **HashSet<T>**: • Search: O(1) on average (O(n) in worst-case for hash collisions) • Insert/Delete: O(1) on average (O(n) in worst-case for hash collisions) 13. **LinkedList<T>**: • Access: O(n) • Search: O(n) • Insert/Delete (any position): O(1) with a reference 14. **Queue<T>**: • Enqueue (Insert): O(1) • Dequeue (Delete): O(1) • Peek (Access front): O(1)
Objects	Yes, C# supports object-orientation.

If this language support object-orientation, provide an example of how you would write a simple object with a default constructor and then how you would instantiate it.

```
public class Person
{
   // Properties
   public string Name { get; set; }
   public int Age { get; set; }

   // Default constructor
   public Person()
   {
      Name = "John Doe";
      Age = 30;
   }
}//end constructor

class Program
{
   static void Main(string[] args)
   {
      // Instantiate a Person object using the default constructor
      Person person1 = new Person();

      // Access the properties of the person object
      Console.WriteLine("Name: " + person1.Name);
      Console.WriteLine("Age: " + person1.Age);
   }
}//end class
```

Runtime Environment

What runtime environment does this language compile to? For example, Java compiles to the Java Virtual Machine.
Do other languages also compile to this runtime? If so, what these other languages?

C# code is typically compiled to the Common Language Runtime (CLR), which is a part of the Microsoft .NET framework. The CLR provides a runtime environment for executing compiled C# code. C# applications are compiled into Intermediate Language (IL) code, which is then Just-In-Time (JIT) compiled into machine code by the CLR at runtime. The CLR is a key component of the .NET framework and provides features such as memory management, garbage collection, security, and cross-language integration.
Other languages that target the CLR include Visual Basic .NET (VB.NET), F#, and more. These languages can interoperate seamlessly within the .NET ecosystem, as they all share the same runtime environment.

Libraries/Frameworks

What are the popular libraries or frameworks used by programmers for this language? List at least three (3) and describe what they are used for.

Three popular libraries and frameworks used by programmers for C# include:

1. **.NET Framework / .NET Core / .NET 5 and Later**:

	• Provides a comprehensive set of libraries and frameworks for building a variety of applications, including desktop, web, cloud, mobile, and IoT applications. It offers a standardized way to develop, build, and deploy C# applications. • Use Cases: • Building Windows desktop applications (Windows Forms, WPF). • Developing web applications using ASP.NET. • Creating cross-platform and cloud-native applications with .NET Core or .NET 5+. • Writing libraries that can be used across different .NET platforms. • Mobile app development with Xamarin. 2. **Entity Framework**: • Simplifies database access and data manipulation in C# applications. It allows developers to work with databases using high-level object-oriented concepts, making database interaction more intuitive and productive. • Use Cases: • Building data-driven applications and services. • Mapping database tables to C# objects. • Querying and updating data with LINQ (Language-Integrated Query). • Providing automatic database schema generation and migration. • Supporting various database backends, such as SQL Server, MySQL, and SQLite. 3. **ASP.NET Core**: • An open-source, cross-platform framework for building web applications. It's a part of the .NET ecosystem and offers a modern, high-performance, and modular platform for creating web APIs and web applications. • Use Cases: • Developing RESTful web services and APIs. • Building web applications and Single Page Applications (SPAs) using technologies like Razor Pages and Blazor. • Supporting cross-platform development for both Windows and non-Windows environments. • Implementing real-time features using SignalR. • Deploying applications as Docker containers or to cloud platforms like Azure, AWS, and Google Cloud.
Domains *What industries or domains use this programming language? Provide at least three specific examples of companies that use this language and what they use it*	1. **Finance and Banking**: • **JPMorgan Chase**: Uses C# to develop trading and risk management systems, as well as for building analytical tools. 2. **Gaming**:

for. ***E.g. Company X uses C# for its line of business applications.***	• **Electronic Arts (EA)**: Uses C# in game development, commonly used in conjunction with the Unity game engine, which provides a C# scripting environment. Developers at EA use C# for game logic, physics simulations, and user interface development. 3. **Healthcare and Medical Devices**: • **Siemens Healthineers**: Uses C# in medical technology, for developing software applications for medical imaging and diagnostics. Includes creating user interfaces, data processing, and controlling medical devices.