নিশ্চয়ই! নিচে C# ইন্টারভিউতে সাধারণভাবে জিজ্ঞাসিত কয়েকটি প্রশ্ন এবং তাদের বাংলা উত্তর দিলাম:

**১. C# কী?**

**উত্তর:**  
C# হলো মাইক্রোসফটের তৈরি একটি আধুনিক, অবজেক্ট-ওরিয়েন্টেড প্রোগ্রামিং ভাষা যা .NET ফ্রেমওয়ার্কের উপর চলে। এটা সাধারণত উইন্ডোজ অ্যাপ্লিকেশন, ওয়েব সার্ভিস, গেম ডেভেলপমেন্ট এবং আরও অনেক কাজের জন্য ব্যবহৃত হয়।

**২. Value Type আর Reference Type এর মধ্যে পার্থক্য কী?**

**উত্তর:**

* **Value Type:** ডেটা সরাসরি স্টোর করে। উদাহরণ: int, float, bool, struct ইত্যাদি।
* **Reference Type:** ডেটার ঠিকানা (reference) স্টোর করে, ডেটা মেমরিতে আলাদা থাকে। উদাহরণ: class, array, string ইত্যাদি।

**৩. What is a delegate in C#? (C#-এ delegate কী?)**

**উত্তর:**  
Delegate হলো এমন একটি টাইপ যা মেথডের রেফারেন্স ধরে রাখতে পারে এবং সেই মেথড কল করতে পারে। এটা ফাংশন পয়েন্টারের মতো কাজ করে।

**৪. Difference between abstract class and interface?**

**উত্তর:**

* **Abstract class:** কিছু বেসিক ইমপ্লিমেন্টেশন থাকতে পারে, এবং ইনহেরিট করা যায়। এক ক্লাস শুধুমাত্র একবার ইনহেরিট করতে পারে।
* **Interface:** সম্পূর্ণ অ্যাবস্ট্রাক্ট, কোনো ইমপ্লিমেন্টেশন নেই, একাধিক ইন্টারফেস ইমপ্লিমেন্ট করা যায়।

**৫. What is boxing and unboxing in C#?**

**উত্তর:**

* **Boxing:** Value type কে reference type এ কনভার্ট করা।
* **Unboxing:** Reference type থেকে value type এ কনভার্ট করা।

**৬. What is Garbage Collection in .NET?**

**উত্তর:**  
Garbage Collector হলো .NET এর মেমরি ম্যানেজমেন্ট সিস্টেম যা অপ্রয়োজনীয় অবজেক্টগুলোকে মেমরি থেকে ডিলিট করে দেয়।

**৭. Explain access modifiers in C#.**

**উত্তর:**

* **public:** যেকোনো জায়গা থেকে অ্যাক্সেস করা যায়।
* **private:** শুধুমাত্র একই ক্লাসের ভিতর থেকে অ্যাক্সেস করা যায়।
* **protected:** শুধুমাত্র একই ক্লাস এবং ডেরাইভড ক্লাস থেকে অ্যাক্সেস করা যায়।
* **internal:** একই অ্যাসেম্বলির মধ্যে অ্যাক্সেস করা যায়।
* **protected internal:** একই অ্যাসেম্বলির মধ্যে অথবা ডেরাইভড ক্লাস থেকে অ্যাক্সেস করা যায়।

**৮. What is async and await in C#?**

**উত্তর:**  
async এবং await keywords ব্যবহার করে আপনি asynchronous প্রোগ্রামিং করতে পারেন, যা কোডকে non-blocking এবং efficient করে তোলে।

**৯. Explain partial class in C#.**

**উত্তর:**  
Partial class হলো এমন একটি ক্লাস যা বিভিন্ন ফাইল থেকে ডিফাইন করা যায় এবং কম্পাইলার একত্রে মিলিয়ে একটি ক্লাস হিসেবে বিবেচনা করে।

**১০. Difference between String, StringBuilder?**

**উত্তর:**

* **String:** Immutable অর্থাৎ পরিবর্তন করলে নতুন স্ট্রিং তৈরি হয়।
* **StringBuilder:** Mutable, একই অবজেক্টে স্ট্রিং পরিবর্তন করতে পারে, তাই যখন অনেক স্ট্রিং অপারেশন দরকার তখন StringBuilder ভালো।

অবশ্যই! এবার আরও বিস্তারিতভাবে C# ইন্টারভিউ প্রশ্ন ও উত্তর বাংলায় উপস্থাপন করছি:

**🟠 ১. C# কী এবং কেন ব্যবহার করা হয়?**

**উত্তর:**  
C# (উচ্চারণ: সি-শার্প) হলো একটি আধুনিক, অবজেক্ট-ওরিয়েন্টেড প্রোগ্রামিং ভাষা যা মাইক্রোসফট তৈরি করেছে .NET প্ল্যাটফর্মের জন্য।  
এই ভাষাটি Java এর অনুরূপ হলেও .NET ফ্রেমওয়ার্কের জন্য বিশেষভাবে ডিজাইন করা হয়েছে। এটি ব্যবহার করা হয়:

* ডেস্কটপ অ্যাপ্লিকেশন (Windows Forms, WPF)
* ওয়েব অ্যাপ্লিকেশন (ASP.NET)
* গেম ডেভেলপমেন্ট (Unity)
* ক্লাউড অ্যাপ্লিকেশন (Azure)
* মোবাইল অ্যাপ্লিকেশন (Xamarin)

🔹 **C# এর সুবিধা:**

* টাইপ সেফ (Type Safe)
* মেমোরি ম্যানেজমেন্ট অটোমেটিক (Garbage Collector)
* ল্যাংগুয়েজ ইন্টিগ্রেটেড কোয়েরি (LINQ)
* অ্যাসিঙ্ক্রোনাস প্রোগ্রামিং (async/await)

**🟠 ২. Value Type ও Reference Type এর মধ্যে পার্থক্য কী?**

**উত্তর:**

| **বিষয়** | **Value Type** | **Reference Type** |
| --- | --- | --- |
| ডেটা সংরক্ষণ | Stack-এ | Heap-এ |
| কপি করলে | নতুন কপি তৈরি হয় | একই মেমোরি রেফারেন্স শেয়ার করে |
| উদাহরণ | int, float, bool, struct | class, object, string, array |

🔸 উদাহরণ:

int a = 10;

int b = a; // এখানে b = 10 (new copy)

class Person { public string Name; }

Person p1 = new Person();

Person p2 = p1; // এখানে p2 একই reference ধরে রাখে

**🟠 ৩. Delegate কী এবং কেন দরকার?**

**উত্তর:**  
Delegate হলো এমন একটি টাইপ যেটা মেথডকে রেফারেন্স করতে পারে। একে বলা হয় type-safe function pointer।

🔸 Delegate এর মাধ্যমে:

* ইভেন্ট হ্যান্ডলিং করা যায়
* কলব্যাক মেথড পাঠানো যায়
* স্ট্রাটেজি প্যাটার্ন ইমপ্লিমেন্ট করা যায়

🔸 সাধারণ delegate উদাহরণ:

public delegate void PrintMessage(string message);

void Show(string msg) {

Console.WriteLine(msg);

}

PrintMessage printer = Show;

printer("Hello from delegate!");

**🟠 ৪. Abstract Class vs Interface**

| **বিষয়** | **Abstract Class** | **Interface** |
| --- | --- | --- |
| ইনহেরিটেন্স | একটিই ইনহেরিট করা যায় | একাধিক ইন্টারফেস ইমপ্লিমেন্ট করা যায় |
| মেথড | কিছু মেথডের বডি থাকতে পারে | সব মেথড abstract (C# 8.0 থেকে default method যোগ হয়েছে) |
| ফিল্ড/প্রপার্টি | ফিল্ড, প্রপার্টি, কনস্ট্রাক্টর থাকতে পারে | শুধু মেথড, ইভেন্ট, প্রপার্টির সিগনেচার থাকে |

🔸 কখন Interface ব্যবহার করবো?  
যখন কেবল কোন কাজ করতে হবে তার চুক্তি দিতে চাই।

🔸 কখন Abstract Class ব্যবহার করবো?  
যখন কিছু সাধারণ বাস্তবায়ন (implementation) দিতে চাই।

**🟠 ৫. Boxing ও Unboxing কী?**

**উত্তর:**

* **Boxing:** একটি value type কে object (reference type)-এ কনভার্ট করা।
* **Unboxing:** object থেকে value type-এ কনভার্ট করা।

🔸 উদাহরণ:

int i = 123;

object o = i; // boxing

int j = (int)o; // unboxing

⚠️ Boxing/Unboxing বেশি হলে পারফরম্যান্স খারাপ হয়।

**🟠 ৬. Garbage Collector কী?**

**উত্তর:**  
Garbage Collector (GC) হলো .NET runtime-এর একটি কম্পোনেন্ট যা এমন অবজেক্ট গুলোকে মেমরি থেকে রিমুভ করে যেগুলোর আর কোনো reference নেই।

🔹 এর ফলে:

* মেমরি লিক রোধ করা যায়
* ডেভেলপারকে ম্যানুয়ালি মেমরি ম্যানেজ করতে হয় না

**🟠 ৭. Access Modifiers গুলো কী?**

**উত্তর:**

| **Modifier** | **Access Level** |
| --- | --- |
| public | যেকোনো জায়গা থেকে অ্যাক্সেস করা যায় |
| private | শুধুমাত্র ক্লাসের ভিতরে |
| protected | ক্লাস এবং তার ডেরাইভ ক্লাসে |
| internal | একই অ্যাসেম্বলি এর ভিতরে |
| protected internal | একই অ্যাসেম্বলি বা ডেরাইভ ক্লাসে |
| private protected | একই অ্যাসেম্বলি + ডেরাইভ ক্লাসেই কেবল |

**🟠 ৮. async এবং await কী?**

**উত্তর:**  
C#-এ async ও await asynchronous প্রোগ্রামিং এর জন্য ব্যবহৃত হয়। এর ফলে অ্যাপ্লিকেশন block না করে কাজ চালিয়ে যেতে পারে।

🔸 উদাহরণ:

public async Task<string> GetDataAsync() {

HttpClient client = new HttpClient();

string result = await client.GetStringAsync("https://example.com");

return result;

}

**🟠 ৯. partial class কী?**

**উত্তর:**  
একটি ক্লাসকে একাধিক ফাইলে ভাগ করে লেখা যায় partial keyword ব্যবহার করে।

🔸 এটি সাধারণত বড় বড় ক্লাস বা auto-generated কোড ব্যবহারে করা হয়।

// File1.cs

public partial class MyClass {

public void Method1() { }

}

// File2.cs

public partial class MyClass {

public void Method2() { }

}

**🟠 ১০. String vs StringBuilder**

| **বিষয়** | **String** | **StringBuilder** |
| --- | --- | --- |
| পরিবর্তনযোগ্য | নয় (immutable) | হ্যাঁ (mutable) |
| পারফরম্যান্স | ধীর, যদি অনেক পরিবর্তন করা হয় | দ্রুত |
| ব্যবহারের ক্ষেত্র | ছোট ও কম স্ট্রিং অপারেশনে | বড় লুপে বা অনেক স্ট্রিং কনক্যাটেনেশনে |

🔸 উদাহরণ:

StringBuilder sb = new StringBuilder();

sb.Append("Hello");

sb.Append(" World");

Console.WriteLine(sb.ToString()); // Hello World

অসাধারণ! নিচে **Advanced Level C# Interview Questions and Answers** বাংলায় বিস্তারিতভাবে তুলে ধরা হলো:

**🔶 ১. Dependency Injection (DI) কী এবং এটি কীভাবে কাজ করে C#-এ?**

**উত্তর:**  
Dependency Injection (DI) হলো একটি ডিজাইন প্যাটার্ন যা একটি অবজেক্টের উপর নির্ভরশীল ক্লাসগুলোকে বাইরের কোন সোর্স থেকে ইনজেক্ট করে দেয়।

🔸 মূল উদ্দেশ্য:

* ক্লাসগুলোকে কম কুপলড করা (loosely coupled)
* টেস্টযোগ্য করা
* মেইন্টেনেবল কোড লেখা

🔸 উদাহরণ:

public interface IMessageService {

void Send(string message);

}

public class EmailService : IMessageService {

public void Send(string message) {

Console.WriteLine("Email sent: " + message);

}

}

public class Notification {

private readonly IMessageService \_messageService;

public Notification(IMessageService messageService) {

\_messageService = messageService;

}

public void Notify(string message) {

\_messageService.Send(message);

}

}

👉 .NET Core-এ DI built-in service container এর মাধ্যমে কাজ করে (Startup.cs বা Program.cs এ services.AddScoped<...>() ইত্যাদি)।

**🔶 ২. What is Reflection in C#?**

**উত্তর:**  
Reflection হলো .NET-এর একটি শক্তিশালী ফিচার যার মাধ্যমে রানটাইমে অ্যাসেম্বলি, টাইপ, প্রপার্টি, মেথড, ইত্যাদি সম্পর্কে জানা যায় এবং এক্সেস করা যায়।

🔸 উদাহরণ:

Type type = typeof(string);

MethodInfo[] methods = type.GetMethods();

foreach (var method in methods) {

Console.WriteLine(method.Name);

}

**ব্যবহার:**

* প্লাগিন ডেভেলপমেন্ট
* টেস্টিং ফ্রেমওয়ার্ক
* ORM (Entity Framework)
* JSON Serializers

**🔶 ৩. What is the difference between Task, Thread এবং async/await?**

| **বিষয়** | **Thread** | **Task** | **async/await** |
| --- | --- | --- | --- |
| লো-লেভেল | হ্যাঁ | না | না |
| টাস্ক শিডিউলিং | ম্যানুয়ালি | টাস্ক ম্যানেজার | Compiler-managed |
| ইউজার কন্ট্রোল | বেশি | কিছুটা কম | সবচেয়ে কম |
| ব্যবহারের জায়গা | Background কাজ | Parallel কাজ | Non-blocking UI বা I/O bound কাজ |

// async/await উদাহরণ

public async Task<int> GetDataAsync() {

await Task.Delay(2000); // simulate delay

return 42;

}

**🔶 ৪. What are extension methods?**

**উত্তর:**  
Extension Method এমন একটি static method যা আপনি একটি এক্সিস্টিং টাইপে নতুন method এর মত করে যোগ করতে পারেন, ক্লাসটিকে ইনহেরিট বা পরিবর্তন না করেই।

🔸 উদাহরণ:

public static class StringExtensions {

public static string ToTitleCase(this string str) {

return CultureInfo.CurrentCulture.TextInfo.ToTitleCase(str.ToLower());

}

}

// ব্যবহার:

string name = "john doe";

Console.WriteLine(name.ToTitleCase()); // John Doe

**🔶 ৫. Difference between IEnumerable, ICollection, IQueryable**

| **বিষয়** | **IEnumerable** | **ICollection** | **IQueryable** |
| --- | --- | --- | --- |
| ডেটা enumeration | হ্যাঁ | হ্যাঁ | হ্যাঁ |
| Add/Remove | না | হ্যাঁ | না |
| Query execution | Immediate | Immediate | Deferred |
| LINQ support | In-Memory | In-Memory | Remote (e.g., SQL) |

👉 IQueryable LINQ কে SQL-এ কনভার্ট করে (e.g., Entity Framework), যেখানে IEnumerable মেমরিতে ডেটা আনার পর প্রসেস করে।

**🔶 ৬. What is the difference between readonly vs const vs static readonly?**

| **Keyword** | **Compile Time** | **Runtime** | **Static** |
| --- | --- | --- | --- |
| const | হ্যাঁ | না | হ্যাঁ |
| readonly | না | হ্যাঁ | না |
| static readonly | না | হ্যাঁ | হ্যাঁ |

public const int x = 10; // compile-time constant

public readonly int y; // can set in constructor

public static readonly int z = 100; // can set in static constructor

**🔶 ৭. What is covariance and contravariance in C#?**

**উত্তর:**  
Covariance এবং Contravariance হলো জেনেরিক টাইপ প্যারামিটারে টাইপ কনভার্সন (Inheritance অনুযায়ী) সমর্থন।

**✅ Covariance (output):**

IEnumerable<string> strings = new List<string>();

IEnumerable<object> objects = strings; // Valid (Covariance)

**✅ Contravariance (input):**

Action<object> actObj = s => Console.WriteLine(s);

Action<string> actStr = actObj; // Valid (Contravariance)

**🔶 ৮. What is async streams in C# 8.0?**

**উত্তর:**  
C# 8.0 এ async stream চালু হয়েছে, যেখানে IAsyncEnumerable<T> ব্যবহার করে await foreach দিয়ে asynchronous ভাবে ডেটা stream করা যায়।

public async IAsyncEnumerable<int> GetNumbersAsync() {

for (int i = 0; i < 5; i++) {

await Task.Delay(500);

yield return i;

}

}

await foreach (var number in GetNumbersAsync()) {

Console.WriteLine(number);

}

**🔶 ৯. What is Span এবং Memory?**

**উত্তর:**  
Span<T> এবং Memory<T> হলো high-performance টাইপ যা stack বা heap-এ ডেটা ছাড়াই মেমরির উপর রেফারেন্স দিয়ে কাজ করতে পারে।

* Span<T> → শুধুমাত্র synchronous এবং stack-safe।
* Memory<T> → asynchronous context-এ কাজ করে।

Span<int> numbers = stackalloc int[3] { 1, 2, 3 };

numbers[0] = 42;

**🔶 ১০. What is a memory leak in .NET এবং কীভাবে রোধ করবেন?**

**উত্তর:**  
যখন অবজেক্টগুলোর reference ছাড়া থাকলেও তারা Garbage Collector দ্বারা collect হয় না, তখন তাকে **Memory Leak** বলে।

**✅ রোধের উপায়:**

* Events Unsubscribe না করলে memory leak হয়
* Static references সতর্কভাবে ব্যবহার করতে হবে
* IDisposable ব্যবহার করে resources release করা

public void Dispose() {

// event থেকে unsubscribe

myButton.Click -= OnClick;

}

অসাধারণ অনুরোধ! নিচে C# 9, C# 10 এবং C# 11 এর নতুন ফিচারগুলো বাংলায় **সংক্ষিপ্ত এবং বিস্তারিত ব্যাখ্যা সহ** তুলে ধরা হলো।

**✅ 🆕 C# 9 (2020) এর নতুন ফিচারসমূহ**

**1. Record Types**

**ব্যাখ্যা:** Immutable ডেটা মডেল তৈরির জন্য নতুন record টাইপ।

public record Person(string Name, int Age);

🔸 এটি value-based equality ব্যবহার করে:

var p1 = new Person("A", 25);

var p2 = new Person("A", 25);

Console.WriteLine(p1 == p2); // True

**2. Init-only Properties**

**ব্যাখ্যা:** set শুধু object initializer-এ করা যাবে, runtime-এ নয়।

public class Car {

public string Brand { get; init; }

}

var car = new Car { Brand = "Toyota" };

// car.Brand = "Honda"; // Error!

**3. Top-level Statements**

**ব্যাখ্যা:** Main method না লিখেই সরাসরি কোড চালানো যায়।

Console.WriteLine("Hello, World!");

**4. With-expressions**

**ব্যাখ্যা:** Record ক্লাসের নতুন কপি তৈরি করতে with ব্যবহার।

var p1 = new Person("Rahim", 30);

var p2 = p1 with { Age = 31 };

**5. Pattern Matching Improvements**

* **Relational Patterns:** <, >, <=, >= যোগ হয়েছে
* **Logical Patterns:** and, or, not ব্যবহার করা যায়

if (x is > 0 and < 100)

Console.WriteLine("Valid");

**✅ 🆕 C# 10 (2021) এর নতুন ফিচারসমূহ**

**1. Global Using Directives**

**ব্যাখ্যা:** সব ফাইলে using লিখার দরকার নেই, একবার global হিসেবে লিখলেই হবে।

// GlobalUsings.cs

global using System;

global using System.Collections.Generic;

**2. File-scoped Namespaces**

**ব্যাখ্যা:** কোড কম লেখার জন্য {} ব্লক ছাড়াই namespace।

namespace MyApp.Models;

public class Product {

public string Name { get; set; }

}

**3. Constant interpolated strings**

const string lang = "C#";

const string message = $"Welcome to {lang} 10!"; // Valid!

**4. Record Structs**

public readonly record struct Point(int X, int Y);

**5. Improved Lambda Syntax**

var add = (int x, int y) => x + y;

**✅ 🆕 C# 11 (2022) এর নতুন ফিচারসমূহ**

**1. Raw String Literals**

**ব্যাখ্যা:** Multi-line string বা JSON/XML block লেখার জন্য সহজ ফরম্যাট।

string json = """

{

"name": "Rahim",

"age": 25

}

""";

**2. Generic Attributes**

**ব্যাখ্যা:** Attribute ক্লাস এখন generic হতে পারে।

public class MyAttribute<T> : Attribute { }

[MyAttribute<string>]

public class TestClass { }

**3. List Patterns**

**ব্যাখ্যা:** Pattern matching এখন array বা list-এও কাজ করে।

int[] numbers = [1, 2, 3];

if (numbers is [1, 2, 3]) {

Console.WriteLine("Matched");

}

**4. Required Members**

**ব্যাখ্যা:** কোন প্রপার্টি object initializer এ অবশ্যই সেট করতে হবে।

public class User {

public required string Name { get; set; }

}

var user = new User { Name = "Karim" }; // ✅

var user2 = new User(); // ❌ compile-time error

**5. UTF-8 String Literals (u8)**

ReadOnlySpan<byte> utf8 = "বাংলা"textu8;

**📌 উপসংহার (সারাংশ)**

| **Version** | **ফোকাস** | **মূল ফিচার** |
| --- | --- | --- |
| C# 9 | Immutable Data | Record, init-only, pattern matching |
| C# 10 | Productivity | Global using, file-scoped namespace |
| C# 11 | Expressiveness | Raw string, generic attribute, required |