1. C# چیست و چگونه استفاده می‌شود؟  
   # یک زبان برنامه‌نویسی مدرن و شیء‌گرا است که توسط مایکروسافت به عنوان بخشی از ابتکار .NET توسعه یافته است. این زبان برای توسعه طیف گسترده‌ای از برنامه‌ها از جمله وب، موبایل، دسکتاپ و بازی‌ها طراحی شده است. C# قدرت C++ را با سادگی Visual Basic ترکیب کرده و شامل تایپ قوی، اصول دستوری، اعلانی، تابعی، عمومی، شیء‌گرا (بر پایه کلاس) و برنامه‌نویسی مولفه‌گرا می‌شود. C# به طور معمول برای موارد زیر استفاده می‌شود:
   1. ساخت برنامه‌های ویندوز با فریم‌ورک‌هایی مثل Windows Forms و WPF.
   2. توسعه برنامه‌ها و خدمات وب با استفاده از ASP.NET.
   3. ایجاد برنامه‌های موبایل با Xamarin.
   4. توسعه بازی با استفاده از Unity.
   5. برنامه‌های ابری با استفاده از Azure.
2. ویژگی‌های اصلی C# چیست؟  
   # دارای چندین ویژگی کلیدی است که آن را به یک زبان برنامه‌نویسی قدرتمند و منعطف تبدیل می‌کند:
   1. شیء‌گرا: پشتیبانی از اصولی مانند ارث‌بری، محصورسازی و چندریختی.
   2. تایپ قوی: اعمال ایمنی نوع، کاهش خطاهای زمان اجرا.
   3. مدیریت خودکار حافظه: استفاده از جمع‌آوری زباله (garbage collection) برای مدیریت تخصیص و آزادسازی حافظه.
   4. کتابخانه استاندارد غنی: ارائه مجموعه کاملی از کتابخانه‌ها برای وظایف رایج.
   5. نمایندگان و رویدادها: پشتیبانی از توابع کلاس اول و برنامه‌نویسی مبتنی بر رویداد.
   6. جنریک‌ها: امکان تعریف ساختارهای داده‌ای با ایمنی نوع.
   7. LINQ (زبان یکپارچه پرس و جو): امکان پرس و جوی داده با استفاده از نحو مشابه SQL.
   8. برنامه‌نویسی ناهمگام: پشتیبانی از کلمات کلیدی async و await برای برنامه‌نویسی ناهمگام.
   9. مدیریت استثناها: ارائه مدیریت خطاهای ساختاریافته با استفاده از بلوک‌های try، catch و finally.
3. تفاوت class و object در چیست؟  
   object ها یک نوع در C# به شمار می روند که در واقع نوع پایه محسوب می شود. یعنی تمام کلاس ها و type های مختلف در C# قابلیت cast شدن به نوع object را دارا هستند. در نتیجه، هر کلاسی قابل تبدیل شدن به object را دارد.  
   class ها از سوی دیگر مانند یک قالب یا شکل هستند که میتوان با آنها type های مختلف را در برنامه ایجاد کرد. با class ها می توان type های جدید تعریف کرده و خصوصیت ها و کاربرد های مختلف به آنها نسبت داد..
4. نحوه تبدیل کد Csharp به کد ماشین چگونه است؟  
   در .NET، لایه های مختلفی وجود دارد که کد Csharp را به کد ماشین تبدیل می کنند:
5. تفاوت بین early binding و late binding چیست؟  
   در early binding، کامپایلر در زمان کامپایل (compile time) میتواند تشخیص دهد که object مورد استفاده از چه نوعی بوده و چه method ها و property هایی دارد. مزیت این مدل binding در سرعت بالاتر برنامه و کاهش ارور ها در runtime است.  
   در late binding اما نوع object در زمان اجرا (runtime) مشخص می شود و تا زمان اجرا، نوع آن object مشخص نیست. این روش گرچه انعطاف پذیری برنامه را بالاتر می برد اما در عین حال سرعت اجرا را کمتر و امکان بروز ارور در کد آن هم در زمان اجرا را بالا می برد.
6. تفاوت override و overload چیست؟  
   در overload، امضا یا signature یک متد عوض می شود مانند input parameter ها. اما در override، نحوه پیاده سازی متد تغییر می کند. مثلا نحوه انجام Read() تغییر می کند (خواندن از console، فایل و...)
7. کار checked و unchecked چیست؟  
   برخی کد ها ممکن است در زمان کامپایل، ارور ندهند. به عنوان مثال overflow که در متغیر های عددی بیشتر اتفاق می افتد، در زمان کامپایل اروری به آن داده نمی شود و نیز در زمان اجرا نه تنها به excpetion نمی خورد بلکه یک مقدار متفاوت و نابرابر با مقدار واقعی را نمایش می دهد. با استفاده از اپراتور checked، اگر مواردی مثل overflow اتفاق بیفتد، آنگاه در runtime یک exception اتفاق افتاده و از ادامه برنامه به آن صورت جلوگیری می کند. اپراتور unchecked برعکس آن عمل کرده و کد در runtime دیگر exception بر نمی گرداند. تنظیماتی وجود دارد که میتوان پیش فرض بودن هر کدام از این حالات در کد را در آن تعیین کرد و سپس بر طبق آن از این اپراتور ها در کد استفاده نمود.
8. مزیت BitArray بر آرایه معمولی چیست؟  
   متغیر های از نوع bool گرچه باید فقط یک بیت حافظه اشغال کنند اما در واقعیت در runtime یک بایت را اشغال می کنند. اگه لیستی از چند متغیر یا مقدار bool بخواهیم داشته باشیم، استفاده از BitArray به جای آرایه عادی بهتر است چرا که در آن دقیقا به ازای هر خانه BitArray یک بیت از حافظه به جای یک بایت اشغال می شود.
9. تفاوت Jagged array با Rectangular array در چیست؟  
   در Rectangular، طول آرایه های داخلی یکسان است اما در Jagged طول آرایه های داخلی دلخواه است و در زمان declaration آن می توان این مقدار را به آن داد.
10. تفاوت readonly و const در چیست؟  
    فیلد های readonly در زمان ساخت یک instance از کلاس در constructor قابل مقدار دهی هستند در نتیجه در runtime قابلیت پر شدن دارند. فیلد های const اما در زمان declaration باید مقداردهی شوند در نتیجه در زمان کامپایل باید پر بشوند. همچنین تنها type های bool, string, char, enum و تمام نوع های عددی در خود C# می توانند const باشند.
11. تفاوت boxing و unboxing چیست؟  
    در boxing، یک value-type به reference-type تبدیل می شود و در unboxing، یک reference-type به value-type تبدیل می شود. همچنین در boxing، صرفا ریختن یک value-type در یک reference-type کافی است اما در unboxing باید حتما explicit cast انجام شود و در صورت ناموفق بودن، برنامه exception می دهد.
12. تفاوت typeof و GetType در چیست؟  
    هر دو type یک متغیر یا instance را به ما می دهند اما typeof در زمان کامپایل بررسی و انجام می شود و GetType در runtime.
13. تفاوت class و struct در چیست؟
    1. Class یک reference-type است و struct یک value-type
    2. Struct قابلیت ارث بری ندارد و abstract نیز نمی شود.
    3. Member ها در class به صورت پیش فرض private و در struct به صورت پیش فرض public اند.
14. Callback method چیست؟  
    در chatgpt هست
15. تفاوت object با Generic ها چیست؟  
    generic ها object های type-safe هستند چرا که نوع در زمان کامپایل مشخص شده و دیگر نگران ارور در runtime نخواهیم بود.