# C# Questions

1. **Continue** – lets us skip 1 iteration in loop whereas **break** makes us jump out of loop
2. **Boxing –** process of casting a value type to ref type . **Unboxing –** opposite of boxing.
3. Properties are members of class which are always public and work as wrapper to expose private field of class. They have special methods called accessors. Every property must have atleast one accessor. The properties can be either readonly (only getter), write only (setter) or read/write (both setter and getter).
4. Interface methods don’t have access modifiers because all are without any implementation and are intended to be implemented by the derived class. Interface acts like a contract and helps in loosly coupled design.
5. We can return multiple values from a function using output params, using return only single value can be returned
6. **Dynamic types –** can hold values of any types and their type check happen at runtime
7. **Finally block** – is used to write all clean up code like releasing access to resources closing objects etc.
8. **IDisposable -**  contains a single method dispose to clean all unmanaged resources like db connection objects, files , streams etc
9. **Interface vs Abstract class –** 
   1. A class can implement multiple interface but can inherit only 1 abstract class
   2. A abs class can have non-abstract methods (concreate methods) where interfaces can’t have
   3. A abs class can declare and use vars but interfaces can’t
   4. By default access specifier of interfaces is public
10. Ref is similar to out, both can be used to pass a value type as reference type but ref needs the var to be initialized before passing
11. Read about discards in c#
12. **Overloading**  allows you to make multiple methods in same scope with same name but diff params
13. Overriding allows us to change the implementation of the method
14. **Anonymous types -**  allows to define a obj without defining a type for it so type is generated by compiler and automatically accessible within the current block only
15. **Const** are compiled are run time and need to be initialized where as **readonly** compile at runtime so gets value at runtime
16. **Destructors –** is used to clean up memory and free up the resources. They don’t have access specifiers and cant be inherited and each class can have only 1 destructor
17. **Encapsulation** is implemented via access modifiers in c#
18. Lambda function – anonymous functions which are used to build delegates or expression tree functions
19. **Static constructors –** they are called only the first time class is referenced
20. **We cant override a method in same class**
21. **Serialization** taking array and converting it into stream of btye array.
22. **Compile time polymorphism is known as early binding is achieved using method over loading**
23. **Run time polymorphism is aka late binding is achieved by using method overriding**
24. **Object Pool -**  ready to use objects. It reduces overhead of creating object
25. **Hash tables –** key value pair. It contains value based on the key
26. Abstract classes cannot be initialized.
27. To access a non-abstract method of an abstract class we need to first derive it and create a instance of derived class and then access it from the derived class method.
28. First, compile the source code in the managed code compatible with the C# compiler. Second, combine the above newly created code into assemblies. Third, load the CLR. Last, execute the assembly by CLR to generate the output.
29. **Finalize** method is defined in the object class & is used for cleanup activities. This method is called by GC. GC frees up managed resources automatically but to free up unmanaged resources we have to explicitly call the dispose method
30. **Control** Statements are if, if else, nested if, if else if

# Entity Framework Interview Question

1. Entity framework is an ORM and a persistence framework which allows us to write queries using Linq
2. DbContext class lets us link our entities to db tables via connection string also creates a conn.
3. Micro-ORMs focus on most important task of db and not focus on creating modifying db or change tracking.
4. EF framework is always SQL injection safe as it generates parametrized query.
5. **Proxy object –** the entities generated by EF to support change tracking and lazy loading or POCO are proxy objects. In this class it should be public, not sealed, each property should be virtual, must have public getter setters.
6. **Entity Framework states -**  every entity has to go through various states during its life cycle. These states are defined by enumerator.
   1. **Added**
   2. **Modified**
   3. **Deleted**
   4. **Unchanged**
   5. **Detached**
7. In code first approach POCO are mapped to db tables
8. Migration history tables stores the details about the migrations applied on the database and also the migration class has 2 methods Up() and Down().
9. EF supports automatic migrations so we don’t need to run the migrations when we make changes manually also EF has manual migrations.
10. DbSet is a types entity used to perform CRUD
11. EF allows us to execute Sql Query as well
12. EF supports database access architecture in 2 modes:
    1. Connection Oriented
    2. Disconnected – doesn’t leave the connection open
13. Concurrency is handled by optimistic locking. We can right click on EDMX and select concurrency mode to fixed.
14. context.ContextOptions.LazyLoadingEnabled = false;
15. In Lazy loading we load object on demand rather than all in one go. It is enabled by default.
16. Object Context manages all the database operations, like database connection, and manages various entities of the Entity Model. DB Context is a wrapper around Object Context.
17. Association set define relation between entity sets
18. To increase performance in entity framework we can do following things
    1. Disable change tracking
    2. Use compiled query whenever required
    3. Avoid using views
    4. Fetch required data from database
19. Explicity loading Be default Entity Framework supports Lazy loading, but we can disable the Lazy loading and we can still load the dependent/related entities by calling “Load” method.
20. Storage model, Conceptual Model, and Mapping are the parts of Entity Model.

# LINQ