1. How start the garbage collector?
   1. GC.Collect
2. Params must be?
   1. Last parameter
3. Non Params method takes priority over param method
4. XOR sign
   1. ^
5. Destructor sign
   1. ~
6. IComparable takes parameter as
   1. Object
7. How many parameter a select method can have?
   1. 2
8. Select method is
   1. Extension method of enumerable class
9. List<T> FindAll method return a?
   1. List<T>
10. We can’t declare access modifier in?
    1. Interface
11. Which of these takes any type and any parameter?
    1. Public static void Hole(params object[ ] paramList)
12. How to declare a specific generic type?
    1. public class PrintableCollection<T> where T : IPrintable
13. Which of this is covariant?
    1. interface IRetrieveWrapper<out T>{ T GetData();}
14. Which of this is contravariant?
    1. public interface IComparer<in T>{ int Compare(T x, T y);}
15. How can we convert LINQ to PLINQ?
    1. AsParallel
16. Select method can take?
    1. Two parameter
17. How to subscribe in an event?
    1. +=
18. How to declare an event?
    1. event delegateTypeName eventName
19. Lambda expressions were added to the C# language in version?
    1. 3.0
20. How many elements a lamda expression can contain?
    1. Two elements
21. Which class implements a last-in, first-out mechanism?
    1. *Stack<T>*
22. Which can take keyvalue pair?
    1. SortedList<T>
23. What are collections?
    1. Generic type
24. Declaration of index in interface?
    1. bool this [ int index ] { get; set; }
25. Which class cannot be base class?
    1. sealed
26. Which method is the last implementation?
    1. sealed method
27. Which an interface can’t contain?
    1. Field
28. Which keyword We use to define an extension method?
    1. static
29. If A class derived from another class B, it can access the protected class members of class B is effectively?
    1. Public
30. A class can inherit at most?
    1. One class
31. Using new keyword
    1. new method declare
32. Root of all class?
    1. System.object
33. Which class must have to instantiate?
    1. Sealed class
34. New keyword allocates memory?
    1. From heap
35. Params can take?
    1. One dimensional array
36. Which can encapsulate a set of value?
    1. Indexer
37. What is the difference between abstract and virtual method?
    1. abstract method can not contain body
38. Which keyword define an iteration should return?
    1. yield
39. Which operator split the task?
    1. await
40. Which is a query operator?
    1. from cust in customers select cust.FirstName
41. Which is the sign of lambda?
    1. =>

D**escriptive**

1. Five points of params array page no. 254

2 Important rules of polymorphic method -------- 272

3. Restriction of interface ----------- 293,294

Q1. Answer:

There are several points worth noting about params arrays:

■■ We can’t use the params keyword with multidimensional arrays.

■■ We can’t overload a method based solely on the params keyword

■■ We’re not allowed to specify the ref or out modifier with params arrays.

■■ A params array must be the last parameter

■■ A non-params method always takes priority over a params method.

Q2.Answer:

There are some important rules We must follow when declaring polymorphic methods :

■■ A virtual method cannot be private.

■■ The signatures of the virtual and override methods must be identical; they must have the same name, number, and types of parameters. In addition, both methods must return the same type.

■■ We can only override a virtual method.

■■ If the derived class does not declare the method by using the override keyword, it does not override the base class method.

■■ An override method is implicitly virtual and can itself be overridden in a further derived class.

Q3. Answer:

Interface restrictions :

■■ We’re not allowed to define any fields in an interface, not even static fields.

■■ We’re not allowed to define any constructors in an interface. A constructor is also considered to be an implementation detail of a class or structure.

■■ We’re not allowed to define a destructor in an interface.

■■ We cannot specify an access modifier for any method.

■■ We cannot nest any types (such as enumerations, structures, classes, or interfaces) inside an interface.

■■ An interface is not allowed to inherit from a structure or a class, although an interface can inherit from another interface. Structures and classes contain implementation; if an interface were allowed to inherit from either, it would be inheriting some implementation.