**1. Which of the C# features listed below allow you to customize the behavior of an existing class? Enter the two best answers below.**

a) Polymorphism

b) Class inheritance

c) Interfaces

d) Delegates and events

e) Enumerations

Ans: a) Polymorphism b) Class inheritance

**2. What are the essential differences between C# consts and readonly fields?**

Ans: A field declared as const cannot be changed as it is a compile time constant, and it is loaded in the stack. Const fields can be declared inside the method. In const fields, we can only assign values in declaration part. A field declared as readonly can be changed at runtime using a constructor, and is loaded in the heap. Readonly fields cannot be declared inside the method. In readonly fields, we can assign values in declaration, and in the constructor part.

**3. Write a C# method called Swap that takes two ints as arguments (x and y) and swaps their values without using a third variable. Then, write some code that declares variable “a”,and “b”as int. Initialize their values, and then calls Swap(a, b). After calling Swap, a should have b’s original value and b should have a’s original value. (Hint: ByRef)**

Ans:

public class World

{

public static void **Main**(string[] args)

{

int a=100; int b=50;

Console.WriteLine("Before swap a= "+a+" b= "+b);

**Swap**( ByRef a, ByRef b);

Console.WriteLine("After swap a= "+a+" b= "+b);

}

public static void **Swap**(ByRef int x, ByRef int y){

int a=x; int b=y;

x=x\*y; y=x/y; x=x/y;

}

}

**4. What’s the implicit name of the parameter that gets passed into the class's set method?**

Ans: The implicit name of the parameter that gets passed into the class set method is **value**. The datatype of the parameter value is the datatype of the property

**5. How do you inherit from a class in C#?**

Ans: To inherit from a class in C# use class ChildClassName : ParentClassName.

For Example: class Car: Vehicle Here Car is the child class (derived class), and Vehicle is the parent class (base class)

**6. Does C# support multiple inheritance?**

Ans: No. For example, we have a class **Fly**, and it is inherited by **Vehicles** and **Birds**. Both **Airplane** and a **Bird** can fly but they are not from the same family.

interface A { }

interface B { }

class Base { }

class AnotherClass { }

Ways to inherit:

class SomeClass : A, B { } // from multiple Interface(s)

class SomeClass : Base, B { } // from one Class and Interface(s)

This is not legal:

**class SomeClass : Base, AnotherClass { }**

**7. When you inherit a protected class-level variable, who is it available to?**

Ans: A protected variable in a base class being inherited will be available to the derived class.

**8. Are private class-level variables inherited?**

Ans: NO, they are not inherited.

**9. Describe the accessibility modifier Protected Internally.**

Ans: A protected internal member of a base class is accessible within the class, and assembly containing the base class. It is also accessible in a derived class located in another assembly only if the access occurs through a variable of the derived class type. For example, we have two assembly classes’ Assembly1.cs and Assembly2.cs, and assembly1.cs has a base class with protected internal member and a test class and assembly2.cs has a derived class that extends the base class of assembly1.cs. Now the protected internal member in the base class of assembly1.cs can be accessed by instantiating it in the test class that is within the assembly1.cs and can be accessed from the derived class in the assembly2.cs but the object instantiation must be thorough the variable of the derived class.

**10. C# provides a default constructor for me. I write a constructor that takes a string as a parameter, but also want to keep the no parameter constructor. How many constructors should I write?**

Ans: You must have two constructors. One with the string parameter and other constructor with empty parameter.