C# Mid Sample
By Siam

1. Which of the following keywords is used to define a derived class in C#?

a) `base`
b) `extends`
c) `inherits`
d) `:`

---

2. What is the key characteristic of polymorphism in C#?

a) A class can have multiple constructors.
b) A method can behave differently based on the object calling it.
c) A class can inherit from multiple base classes.
d) A class can be declared abstract.

---

3. Which of the following is true about encapsulation in C#?

a) It restricts access to the inner workings of a class.
b) It allows classes to inherit methods from other classes.
c) It involves declaring methods as `abstract`.
d) It makes all class members static.

1. What does the `params` keyword do in C#?

a) It allows a method to accept a variable number of arguments of the same type.
b) It allows a method to accept only a fixed number of arguments.
c) It is used to declare a parameter as optional.
d) It forces the caller to specify arguments explicitly.

---

2. What is a jagged array in C#?

a) An array that stores elements of different types.
b) An array of arrays, where each element is an array itself.

c) An array that has a fixed size.
d) An array that is automatically resized.

---

3. Which of the following is used to convert a string to an integer in C#?

a) `int.Parse()`
b) `Convert.ToInt32()`
c) `Parse.Int()`
d) Both a and b

```
int[] arr = { 1, 2, 3, 4, 5 };
int result = 1;
for (int i = 0; i < arr.Length; i++)
{
    for (int j = 0; j < arr.Length; j++)
    {
        if (i != j)
        {
            result *= arr[i] + arr[j];
        }
    }
}
Console.WriteLine(result);
```

---

4. What will be the output of the code?

a) 64800
b) 12000
c) 72000
d) 57600

---

5. What is the final value of the `result` variable after the code executes?

a) 64800
b) 12000
c) 57600
d) 72000

---

6. How many times is the `if (i != j)` condition true during the execution of the code?

a) 20
b) 25
c) 16
d) 24

---

7. What would happen if the condition `if (i != j)` were removed?

a) The result would be zero.
b) The result would be the product of all the array elements.
c) The result would be much higher than the current output.
d) The result would be the sum of all the array elements.

```
int x = 7;
int y = 5;
int z = 0;
for (int i = 0; i < 4; i++)
{
    x += y;
    y -= 2;
    z += x;
}
Console.WriteLine(x);
Console.WriteLine(y);
Console.WriteLine(z);
```


7. What will be the output of the code?

a) 25, -1, 90
b) 27, -3, 90
c) 29, -1, 70
d) 25, -3, 70

---

8. What is the final value of `x`?

a) 25
b) 27
c) 29
d) 30

---

9. What is the final value of `z`?

a) 90
b) 80
c) 70
d) 60

---

10. Which of the following is the correct way to define a method in C#?

a) `public void methodName() { }`
b) `void methodName { }`
c) `public void methodName[] { }`
d) `methodName void public() { }`

---

11. Which of the following is the correct way to declare a variable in C#?

a) `int x = 5;`
b) `5 int x;`
c) `int x; = 5`
d) `declare int x = 5;`

---

12. What is the correct way to handle user input in C#?

a) `input.Read()`
b) `Console.ReadInput()`
c) `Console.ReadLine()`
d) `Scanner.ReadLine()`

---

13. Which of the following is NOT a valid data type in C#?

a) `int`
b) `boolean`
c) `long`
d) `bool`

---

14. Which of the following operators is used for comparison in C#?

a) `==`
b) `=`
c) `<=`
d) `&&`

---

15. What does the `break` statement do in a loop?

a) It stops the program execution
b) It terminates the loop and continues with the next statement after the loop
c) It restarts the loop
d) It skips the current iteration

---

16. How do you comment a block of code in C#?

a) `/* comment */`
b) `<!-- comment -->`
c) `# comment`
d) `// comment`

---

17. Which of the following statements is correct about arrays in C#?

a) Arrays can only hold one type of data.
b) Arrays are declared using `Array[]`.
c) Arrays in C# can hold multiple types of data.
d) Arrays do not require initialization.

---

18. What is the output of the following code?

```
int a = 8, b = 3;
Console.WriteLine(a % b);
```

a) 5
b) 2

c) 1
d) 0

---

19. Which of the following keywords is used to define a constant in C#?

a) `final`
b) `static`
c) `const`
d) `readonly`

---

20. What is the purpose of the `static` keyword in C#?

a) To declare a method that is bound to an instance
b) To declare a method or property that is shared across all instances of a class
c) To declare a method that cannot be inherited
d) To declare a method that can be overridden in a subclass

---

21. Which keyword is used to define a class member that can only be modified during object initialization or inside a constructor?

a) `const`
b) `readonly`
c) `static`
d) `private`

---

22. What is the effect of the `virtual` keyword in C#?

a) It defines a method that can only be used once.
b) It allows a method to be overridden in a derived class.
c) It makes a method static.
d) It prevents a method from being overridden.

---

23. What does the `new` keyword do in C#?

a) It creates a new object of a class
b) It creates a new instance of a variable
c) It is used for method overloading
d) It prevents a method from being overridden

26. What does the `this` keyword refer to in C#?

a) The current instance of the class
b) The current method being executed
c) A reference to a global variable
d) A reference to the static class

---

27. What does the `override` keyword do in C#?

a) It modifies a method in the base class.
b) It replaces the functionality of a base class method in a derived class.
c) It prevents method overloading.
d) It makes a method virtual.

---

29. What does the `sealed` keyword do in C#?

a) It prevents a class from being inherited.
b) It prevents a method from being overridden.
c) It locks an object for multi-threaded operations.
d) It hides the class constructor.

---

30. What is the function of the `public` keyword in C#?

a) It makes a variable or method accessible only within its class.
b) It makes a variable or method accessible throughout the entire program.
c) It hides the variable or method from other classes.
d) It makes the variable or method available to derived classes only.

These questions will test both your knowledge of **basic syntax** and your ability to **understand C# keywords** and **complex code outputs**!

Output Tracing :

1.

```
using System;

class Program
{
   static void Main()
   {
      int[][] jaggedArray = new int[4][];

      jaggedArray[0] = new int[] { 1, 2, 3 };
      jaggedArray[1] = new int[] { 4, 5 };

      for (int i = 2; i < jaggedArray.Length; i++)
      {
         jaggedArray[i] = new int[i + 1];

         for (int j = 0; j < jaggedArray[i].Length; j++)
         {
            jaggedArray[i][j] = (i + 1) * (j + 1);
         }
      }

      jaggedArray[1][1] = 100;
      jaggedArray[2][0] = 50;

      Console.WriteLine("Values in jagged array where the row index is even:");
      for (int i = 0; i < jaggedArray.Length; i++)
      {
         if (i % 2 == 0)
            for (int j = 0; j < jaggedArray[i].Length; j++)
            {
               Console.Write(jaggedArray[i][j] + " ");
            }
            Console.WriteLine();
         }
      }

      Console.WriteLine("\nSpecific values in jagged array:");
      for (int i = 0; i < jaggedArray.Length; i++)
      {
         for (int j = 0; j < jaggedArray[i].Length; j++)
         {
            if (jaggedArray[i][j] % 2 == 0)
            {
               Console.WriteLine($"Element at [{i}][{j}] = {jaggedArray[i][j]}");
            }
         }
      }
   }
```

2.

```
using System;
class Animal
{
    public int age, energyLevel;
    protected string type;
    private string sound;
    public Animal(int age, string type) { this.age =
age; this.type = type; energyLevel = 100; }
    public virtual void Speak() {
Console.WriteLine(type + " says: " +
(string.IsNullOrEmpty(sound) ? "Generic Sound" :
sound)); }
    public void Rest() { energyLevel = 100;
Console.WriteLine(type + " is resting."); }
    public virtual void Move() { if (energyLevel > 10)
{ Console.WriteLine(type + " is moving!");
energyLevel -= 10; } else Console.WriteLine(type +
" needs rest."); }
    public void Eat() { energyLevel = Math.Min(100,
energyLevel + 20); }
}
class Mammal : Animal
{
    private string furColor;
    protected bool isPregnant;
    public Mammal(int age, string furColor) :
base(age, "Mammal") { this.furColor = furColor; }
    public override void Speak() {
Console.WriteLine("Mammal says: " + (isPregnant ?
"Soft Growl" : "Growl")); }
    public void Mate(Mammal other) { if
(!isPregnant) { isPregnant = true;
Console.WriteLine("Mammal is pregnant!"); } }
    public void ShowFurColor() {
Console.WriteLine("Fur color: " + furColor); }
}
class Dog : Mammal
{
    private string breed;
    private bool isBarking;
    public Dog(int age, string furColor, string breed) :
base(age, furColor) { this.breed = breed; }
    public override void Speak() {
Console.WriteLine(isBarking ? "Dog barks!" : "Dog
is silent."); }
    public void Bark(bool barkNow) { isBarking =
barkNow; }
    public void PlayFetch() { Eat(); Move(); }
    public void CheckHealth() { if (energyLevel < 30)
{ Console.WriteLine("Dog needs rest."); Rest(); } }
}
```

```
class Program
{
    static void Main()
    {
        Dog dog = new Dog(3, "Brown", "Golden
Retriever");
        dog.Speak();
dog.Bark(true);
dog.Speak();
        dog.PlayFetch();
dog.CheckHealth();
  Mammal mammal = new Mammal(4, "Gray");
        mammal.Mate(dog);
dog.Move();
dog.Rest();
dog.CheckHealth();
    }
}
```

Code Writing :

1. Book Class:

The `Book` class should represent a book in the library. It should have the following private fields:

- `title` (string) — the title of the book.
- `author` (string) — the author of the book.
- `isbn` (string) — the ISBN number of the book.
- `availableCopies` (int) — the number of available copies in the library.

Tasks:

- Implement **properties** for each of these fields.
- Use the following conditions:
  - `availableCopies` should never be set to a negative number.
  - The `isbn` should always be exactly 13 characters long when set.

Additionally, implement the following methods:

- **CheckOut()**: This method should decrease the number of available copies when a book is checked out, but only if there are copies available.
- **ReturnBook()**: This method should increase the available copies when a book is returned.

2. Member Class:

The `Member` class should represent a member of the library. It should have the following fields:

- `memberId` (int) — unique member ID.
- `name` (string) — member's full name.
- `membershipType` (string) — the type of membership (e.g., Regular, Premium).

Tasks:

- Implement properties for `memberId` (only `get`), `name`, and `membershipType`.
- `name` should not be empty, and `membershipType` should be either "Regular" or "Premium". If it's anything else, set it to "Regular".

Then , You need to implement two types of members:

- **RegularMember** (inherits from `Member`) — can borrow up to 5 books.
- **PremiumMember** (inherits from `Member`) — can borrow up to 10 books.

Each type of member should have a method to **borrow a book**:You will need to ensure that the methods check the maximum number of books that can be borrowed for each member type.

In the **Main** function, you will need to:

1. **Create instances** of `Book`, `RegularMember`, and `PremiumMember` classes.
2. **Use the properties** to set the values of books and members.
3. Call the `CheckOut()` and `ReturnBook()` methods to test how books are checked out and returned.
4. Ensure that the `BorrowBook()` method is working correctly for both `RegularMember` and `PremiumMember`, considering the borrowing limits.
5. Validate that the conditions and restrictions you've set on properties (e.g., valid ISBN, positive copies, membership type) are functioning correctly.