MCQ

1. Select the correct statement:
    1. When a MulticastDelegate contains multiple methods and one throws an exception, then invocation stops, and the exception propagates.
    2. You can inherit from MulticastDelegate to create custom delegate types.
    3. delegate.GetInvocationList() can be used to retrieve all individual delegates from a MulticastDelegate.
    4. inheritance hierarchy of MulticastDelegate is Object → Delegate → MulticastDelegate.
    5. If a MulticastDelegate has a return type, it returns an array of all method results.

- A. 1,3,4
- B. 2,4
- C. 1,4,5
- D. 3,4,5
- Answer: A

---

2. What will be output?

```
static void Main(string[] args) {
    char A = 'K';
    char B = Convert.ToChar(76); //76 is ascii of 'L'
    A++;
    B++;
    Console.WriteLine(A+ " " +B);
    Console.ReadLine();
}
```

- A. M L

- B. U L
- C. L M
- D. A B
- Answer: C

---

3. Which of the following statements best describes the difference between declaring a delegate field and declaring an event?

- A. Declaring an event simply generates a multicast delegate field with no additional restrictions on how client code can assign handlers.
- B. Declaring a delegate field prevents external code from removing handlers, while declaring an event allows both adding and removing handlers.
- C. Declaring a delegate field exposes the invocation list to external code, which can replace or clear it, whereas declaring an event restricts external code to using only += and -= operators.
- D. Declaring an event automatically ensures that only one subscriber can be attached at a time, preventing multiple handlers from being invoked.
- Answer: C

---

4. Which of the following statements are true for the following snippets?

```
//Snippet 1:
ArrayList arr = new ArrayList();
arr.Add(100);
arr.Add("hello");
```

```
//Snippet 2:
List<object> list = new List<object>();
list.Add(100);
list.Add("hello");
```

1. Both ArrayList and `List<object>` store their elements as object references, requiring boxing of value types.
2. ArrayList provides compile-time type safety and prevents adding mixed types.
3. Recommendation is using `List<T>` instead of ArrayList for new development.
4. `List<object>` performs better than ArrayList for most operations due to its generic implementation.
5. Using `List<object>` instead of ArrayList automatically eliminates the need for casting when retrieving elements.
6. Both collections fully support LINQ extension methods.

- A. 1,2,3
- B. 2,3,5
- C. 3,4,6
- D. 1,3,4
- Answer: D

---

5. In C#, what is the accessibility level of a member declared as internal protected?

- A. Accessible only within the same assembly.
- B. Accessible only within the same class and its derived classes.
- C. Accessible within the same assembly OR in derived classes (even in another assembly).
- D. Accessible only to derived classes within the same assembly.
- Answer: C

---

6. You're building a simple order processing system where:
    - Orders must be processed in the exact sequence they arrive (**FIFO**).
    - The system should only keep the **100 most recent orders**.
    - Older orders beyond 100 should be **automatically discarded**.

    - No multi-threading is required.
    - Which collection best fits these needs?

- A. `List<Order>`

```
List<Order> orders = new List<Order>(100);
orders.Add(newOrder);
if (orders.Count > 100) orders.RemoveAt(0);
```

- B. `Queue<Order>`

```
Queue<Order> orders = new Queue<Order>(100);
orders.Enqueue(newOrder);
if (orders.Count > 100) orders.Dequeue();
```

- C. `Dictionary<int, Order>`

```
Dictionary<int, Order> orders = new Dictionary<int, Order>();
orders.Add(orders.Count, newOrder);
if (orders.Count > 100) orders.Remove(orders.Count - 100);
```

- D. `LinkedList<Order>`

```
LinkedList<Order> orders = new LinkedList<Order>();
orders.AddLast(newOrder);
if (orders.Count > 100) orders.RemoveFirst();
```

- Answer: B

---

7. Which of the following statements is true for the given snippets?

```
//snippet 1:
List<int> list1 = new List<int>(3);
list1[0] = 9;
```

```
//snippet 2:
List<int> list2 = new List<int>(3);
list2.Add(1);
list2.Add(2);
list2.Add(3);
list2[2] = 10;
```

A. In Snippet 1, `list1[0] = 9;` will compile but throw `ArgumentOutOfRangeException` at runtime.
B. In Snippet 1, `new List<int>(3);` creates a list with three elements initialized to `0`.
C. In Snippet 2, `list2[2] = 10;` will succeed because `Count` is `3` after adding three elements.
D. In Snippet 2, the capacity stays at `3` as long as no additional items are added.

- Answer: C

---

8. Which of the following statement is correct?

- A. In C#, writing int numbers[]; is valid syntax for declaring an array variable.
- B. The params keyword can be applied to any parameter, even if it is not the last parameter in the method signature.
- C. When you declare an array with int[] values = new int[3]{1,2,3};, specifying the length inside the brackets is optional because the initializer provides it.

- D. The params keyword modifies the parameter name rather than the parameter type.
- Answer: C

---

9. What is the output of the following C# code?

```
using System;
using System.Collections.Generic;

class Program
{
    static void Main()
    {
        Dictionary<string, int> ages = new Dictionary<string, int>()
        {
            { "Alice", 25 },
            { "Bob", 30 }
        };

        ages["Charlie"] = 28;
        ages.Remove("Bob");

        if (ages.TryGetValue("Alice", out int age))
            Console.WriteLine(age);
        else
            Console.WriteLine("Not found");
    }
}
```

- A. `25`
- B. `ages["Charlie"] = 28;` fails at runtime.

- C. `Not found`
- D. Throws `KeyNotFoundException`
- Answer: A

---

10. Which of these code snippets will throw an `InvalidOperationException` with the message "Collection was modified"?

- A.

```
var colors = new List<string> { "Red", "Green" };
foreach (var color in colors.ToArray()) {
    colors.Add(color + "!");
}
```

- B.

```
var queue = new Queue<int>(new[] { 1, 2, 3 });
foreach (var num in queue) {
    if (num > 1) queue.Dequeue();
}
```

- C.

```
var dict = new Dictionary<int, string> { { 1, "A" }, { 2, "B" } };
foreach (var pair in dict) {
    Console.WriteLine(pair.Value);
}
```

- D.

```
var numbers = new List<int> { 1, 2, 3 };
for (int i = 0; i < numbers.Count; i++) {
    if (numbers[i] == 2) numbers.RemoveAt(i);
}
```

- Answer: B
