1. \*\*What is the primary purpose of the Garbage Collector in C#?\*\*

- A) To manage memory allocation

- B) To clean up unused objects and free memory

- C) To allocate memory for new objects

- D) To optimize CPU usage

\*\*Answer: B) To clean up unused objects and free memory\*\*

2. \*\*Which of the following types are managed by the Garbage Collector in C#?\*\*

- A) Value types

- B) Reference types

- C) Both value and reference types

- D) Only arrays

\*\*Answer: B) Reference types\*\*

3. \*\*How does the Garbage Collector determine if an object is no longer needed?\*\*

- A) By checking object size

- B) By counting object references

- C) By analyzing the object's type

- D) By inspecting the object's fields

\*\*Answer: B) By counting object references\*\*

4. \*\*What is a "generation" in the context of C# memory management?\*\*

- A) A level of memory used for storing data

- B) A grouping of objects based on their lifetime

- C) A type of object in memory

- D) A method to allocate memory

\*\*Answer: B) A grouping of objects based on their lifetime\*\*

5. \*\*Which generation is considered the oldest in C# memory management?\*\*

- A) Generation 0

- B) Generation 1

- C) Generation 2

- D) Generation 3

\*\*Answer: C) Generation 2\*\*

6. \*\*How often does the Garbage Collector perform a collection of Generation 0?\*\*

- A) Every time a new object is created

- B) When the system is low on memory

- C) Frequently, because objects in Generation 0 have a short lifespan

- D) Only during application shutdown

\*\*Answer: C) Frequently, because objects in Generation 0 have a short lifespan\*\*

7. \*\*Which method can be used to force a garbage collection in C#?\*\*

- A) `GC.Collect()`

- B) `GC.Run()`

- C) `GC.CollectMemory()`

- D) `GC.FreeMemory()`

\*\*Answer: A) GC.Collect()\*\*

8. \*\*What is the purpose of the `Finalize` method in C#?\*\*

- A) To explicitly free resources

- B) To perform cleanup before an object is collected by the garbage collector

- C) To allocate new memory

- D) To prevent garbage collection

\*\*Answer: B) To perform cleanup before an object is collected by the garbage collector\*\*

9. \*\*Which interface should a class implement to support deterministic finalization in C#?\*\*

- A) `IDisposable`

- B) `IFinalizable`

- C) `IGarbageCollectable`

- D) `IMemoryManager`

\*\*Answer: A) IDisposable\*\*

10. \*\*When should the `Dispose` method be called in a class?\*\*

- A) Automatically during garbage collection

- B) When an object is no longer needed and should release unmanaged resources

- C) When the application starts

- D) After every method call

\*\*Answer: B) When an object is no longer needed and should release unmanaged resources\*\*

11. \*\*What happens if an object is not disposed of properly and has unmanaged resources?\*\*

- A) The object will be immediately garbage collected

- B) The unmanaged resources may not be released until the object is garbage collected

- C) The object will be automatically disposed of

- D) The application will crash

\*\*Answer: B) The unmanaged resources may not be released until the object is garbage collected\*\*

12. \*\*What is the role of the `GCHandle` class in C#?\*\*

- A) To manually handle garbage collection

- B) To manage the lifecycle of objects in memory

- C) To create a strong reference to an object, preventing it from being collected

- D) To allocate memory for objects

\*\*Answer: C) To create a strong reference to an object, preventing it from being collected\*\*

13. \*\*What is a "memory leak" in the context of C# applications?\*\*

- A) When the garbage collector runs too frequently

- B) When an application fails to release memory that is no longer in use

- C) When the application allocates too much memory

- D) When the memory is fragmented

\*\*Answer: B) When an application fails to release memory that is no longer in use\*\*

14. \*\*How can you improve memory management in C# applications?\*\*

- A) By using unmanaged code exclusively

- B) By avoiding the use of the `Dispose` pattern

- C) By implementing proper disposal of resources and minimizing object allocations

- D) By increasing the amount of physical memory

\*\*Answer: C) By implementing proper disposal of resources and minimizing object allocations\*\*

15. \*\*Which class provides methods for handling finalization and resource management in C#?\*\*

- A) `Object`

- B) `GC`

- C) `Disposable`

- D) `Marshal`

\*\*Answer: D) Marshal\*\*

16. \*\*What is the purpose of the `using` statement in C#?\*\*

- A) To manage garbage collection

- B) To create a scope for an object and ensure its `Dispose` method is called automatically

- C) To allocate memory

- D) To release unmanaged resources

\*\*Answer: B) To create a scope for an object and ensure its `Dispose` method is called automatically\*\*

17. \*\*What is a "weak reference" in C#?\*\*

- A) A reference that does not prevent the garbage collector from collecting the object

- B) A reference that holds onto an object until it is explicitly released

- C) A reference that is used for storing large objects

- D) A reference that increases the object’s lifespan

\*\*Answer: A) A reference that does not prevent the garbage collector from collecting the object\*\*

18. \*\*How can you track memory usage in a C# application?\*\*

- A) By using the `GC` class methods

- B) By analyzing the code manually

- C) By using the `MemoryStream` class

- D) By reviewing application logs

\*\*Answer: A) By using the `GC` class methods\*\*

19. \*\*Which type of objects are most likely to be collected in Generation 0?\*\*

- A) Long-lived objects

- B) Short-lived objects

- C) Objects with complex dependencies

- D) Large objects

\*\*Answer: B) Short-lived objects\*\*

20. \*\*What is the role of `GC.WaitForPendingFinalizers` in C#?\*\*

- A) To force garbage collection

- B) To wait for all finalizers to complete before continuing

- C) To optimize memory usage

- D) To track memory leaks

\*\*Answer: B) To wait for all finalizers to complete before continuing\*\*