1. \*\*What does the Task Parallel Library (TPL) primarily provide?\*\*

- A) Single-threaded operations

- B) Parallel and asynchronous programming support

- C) Networking utilities

- D) File I/O operations

- \*\*Answer: B) Parallel and asynchronous programming support\*\*

2. \*\*Which class is used to create and manage tasks in TPL?\*\*

- A) TaskManager

- B) TaskFactory

- C) Task

- D) Thread

- \*\*Answer: C) Task\*\*

3. \*\*What method do you use to start a task?\*\*

- A) Task.Start()

- B) Task.Begin()

- C) Task.Execute()

- D) Task.Run()

- \*\*Answer: D) Task.Run()\*\*

4. \*\*What does the `Task.Run` method return?\*\*

- A) A new thread

- B) A Task object

- C) A TaskFactory object

- D) A ThreadPool object

- \*\*Answer: B) A Task object\*\*

5. \*\*How can you wait for a task to complete?\*\*

- A) task.Wait()

- B) task.Complete()

- C) task.End()

- D) task.Join()

- \*\*Answer: A) task.Wait()\*\*

6. \*\*Which method is used to handle exceptions thrown by a task?\*\*

- A) task.HandleExceptions()

- B) task.Catch()

- C) task.ContinueWith()

- D) task.Exception()

- \*\*Answer: C) task.ContinueWith()\*\*

7. \*\*What is the purpose of `Task.WhenAll`?\*\*

- A) Waits for all tasks to complete and returns a single Task

- B) Runs multiple tasks sequentially

- C) Cancels all tasks

- D) Creates a new Task object

- \*\*Answer: A) Waits for all tasks to complete and returns a single Task\*\*

8. \*\*What does `Task.WhenAny` return?\*\*

- A) The first Task to complete

- B) All completed tasks

- C) The last Task to complete

- D) A new Task object

- \*\*Answer: A) The first Task to complete\*\*

9. \*\*What is a `Task<TResult>` used for?\*\*

- A) Returning a result from a task

- B) Creating a new Task object

- C) Handling task exceptions

- D) Waiting for tasks to complete

- \*\*Answer: A) Returning a result from a task\*\*

10. \*\*What method is used to cancel a task?\*\*

- A) task.Cancel()

- B) task.Abort()

- C) task.CancellationToken.Cancel()

- D) task.End()

- \*\*Answer: C) task.CancellationToken.Cancel()\*\*

11. \*\*What is a `CancellationToken` used for?\*\*

- A) To signal a task to stop executing

- B) To start a new Task

- C) To handle task exceptions

- D) To wait for a task to complete

- \*\*Answer: A) To signal a task to stop executing\*\*

12. \*\*Which method is used to run code after a task completes regardless of its outcome?\*\*

- A) task.ContinueWith()

- B) task.OnCompletion()

- C) task.FollowUp()

- D) task.After()

- \*\*Answer: A) task.ContinueWith()\*\*

13. \*\*What does `Parallel.For` do?\*\*

- A) Runs iterations in parallel

- B) Creates multiple tasks

- C) Waits for tasks to complete

- D) Cancels tasks

- \*\*Answer: A) Runs iterations in parallel\*\*

14. \*\*Which class provides a way to create and manage a pool of threads?\*\*

- A) ThreadPool

- B) TaskManager

- C) TaskFactory

- D) TaskScheduler

- \*\*Answer: A) ThreadPool\*\*

15. \*\*What is the default TaskScheduler used by TPL?\*\*

- A) DefaultScheduler

- B) TaskScheduler.Current

- C) ThreadPoolScheduler

- D) TaskScheduler.Default

- \*\*Answer: D) TaskScheduler.Default\*\*

16. \*\*What is the purpose of `Parallel.ForEach`?\*\*

- A) To execute a foreach loop in parallel

- B) To create multiple tasks

- C) To wait for tasks to complete

- D) To handle task exceptions

- \*\*Answer: A) To execute a foreach loop in parallel\*\*

17. \*\*Which method in TPL can be used to handle a task's result and continue processing?\*\*

- A) task.ContinueWith()

- B) task.Result()

- C) task.OnCompletion()

- D) task.Handle()

- \*\*Answer: A) task.ContinueWith()\*\*

18. \*\*How do you ensure that a task always runs on the thread pool?\*\*

- A) Use Task.Run()

- B) Use TaskFactory.StartNew()

- C) Use TaskFactory.Start()

- D) Use ThreadPool.QueueUserWorkItem()

- \*\*Answer: A) Use Task.Run()\*\*

19. \*\*Which property of a Task indicates whether the task has been canceled?\*\*

- A) IsCanceled

- B) Cancelled

- C) IsCompleted

- D) HasCanceled

- \*\*Answer: A) IsCanceled\*\*

20. \*\*How can you create a task that runs on a specific `TaskScheduler`?\*\*

- A) Use Task.Factory.StartNew() with the TaskScheduler parameter

- B) Use Task.Run() with the TaskScheduler parameter

- C) Use Task.Create() with the TaskScheduler parameter

- D) Use Task.Begin() with the TaskScheduler parameter

- \*\*Answer: A) Use Task.Factory.StartNew() with the TaskScheduler parameter\*\*