Links Referred:

<https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming>

1. What is Task?  
   Task is an object that represents an ongoing process which will be completed in future.  
   Used to run program asynchronously.
2. What are ways to create task?  
   Task.Factory.StartNewTask(Delegate)  
   Task.Run(Delegate)  
   Task task = new Task()
3. Why we need task? <https://blog.slaks.net/2013-10-11/threads-vs-tasks/>
   1. A task resembles a thread or [ThreadPool](https://docs.microsoft.com/en-us/dotnet/api/system.threading.threadpool) work item, but at a higher level of abstraction.
   2. Task is more abstract then threads.
   3. Task is are generally created on the thread pool which are treated as background threads while thread is by default not background
   4. **Thread Does Not Return Results**
   5. **No Continuation in Thread**
   6. **Cancellation**
   7. **Exception : Better exception handling capabilities. Exception can be handled at parent level as well**
4. Difference between various ways of creating task?
5. How to cancel an ongoing task?
   1. Use CancellationTokenSource and CancellationToken
6. Can you continue one task after one task? **ContinueWith**
7. How many ways one can wait for Task to complete?
   1. Task.Wait, Task.WaitAny, Task.WaitAll
8. How task works internally?
9. How task improves performance in Web Application?
10. Exception Handling in Task? AggergateException
    1. Can we catch exception in parent method try catch?  
       Yes, using Status property
11. When we have task then why one should use async/await?
12. How long running task is scheduled by task scheduler on thread pool?
13. What is threadpool?
14. What are task creation Options?
    1. AttachedToParent
    2. DenyChildAttach
    3. HideScheduler
    4. LongRunning
    5. None
    6. PreferFairness
15. What are task continuationOptions?
16. What is parallel.foreach and how it works?