Service Exercise

The goal of this exercise is to implement a service that handles requests using connections.

You will need to create a class that implements the interface IService. This class will receive a stream of requests from clients and will execute each request on a remote server using a connection.

A connection can only handle a single command at a time, and so in order to improve the computation time of the requests, the service should create more than one connection. The maximum number of connections is in a constant named CONNETION\_COUNT.

Each command returns an integer value, once all requests have been processed, the summary of all these integers must be returned.

The interface IService contains the following methods:

1. sendRequest – a client uses this function to send a request to be processed. This method should not block, even if the service is busy processing requests.
2. getSummary – This function will be called after all requests were sent. returns the summary of all integers from all commands that were sent to the service. If there are requests waiting to be processed, then this function will block until they are processed. No requests will be sent after this function is called.

The following classes are given to you, and do not require any change:

1. Connection - simulates a connection to a remote server and allows a command to be run on the remote server. You are also given the rest of the logic around the service.
2. Client – simulates a client that sends requests.
3. Request – This class represents each request from the client. The value inside the Command should be sent to the remote server using a connection.

The only allowed code changes to existing classes are to Program.cs in the main function, to create an instance of your service.

Good Luck!