My Approach to the Pandell Random Number List Task:

I thought I would step through some of my thought processes as I develop a solution to this problem to help you understand how I approached this problem, as I work through it. I will try and be as concise as possible and list things in point form

Requirements:

* Generate a list of random numbers between 1 -10000, which are inclusive, and unique (making it inclusive, and unique are the most interesting parts, and puts it beyond a basic “Fizzbuzz” test)
* I evaluated that the main goals were clearly laid out. That this is more a test of how I can approach this problem, what resourcefulness can I show, while producing something that is tidy, efficient and documented
* No real core elements other than those around the number generation and showing off my programming ability are required so I will strive to produce the expected results, but wrap an interface around producing these results, possibly expanding on the requirements if it doesn’t take too much time (as I’m sure that is part of the evaluation as well, this is a random number generator, not sending someone to the moon) so there is a line between extra features and over engineering. I will try to use my standard coding style with my normal level of documentation that I use in my day to day work.
* Provide the source code, an executable version, (with a “Readme.txt” file attached)

Design:

* No laid out design requirements other than to use C# as a language (and the random number portion)
* design a simple interface via a console application with a small menu, which will cycle letting a user generate multiple lists, the program will stay open until the user chooses to exit
* expand on the requirements allowing users to chose the size of the list, allow them to choose the Maximum number for the list to generate, 10,000 can be input to meet the requirements of the assignment but I will see if I can go as high as the maxvalue of an Int (for extra challenge)
* Also it would most likely useful to output these numbers to the screen, and as well some more long term method, so I will provide an option to let the user to save the output of the list to a file.
* Build in a small help display for a customer who is unfamiliar with how the program works as a menu option
* Consider how to implement the random list, consider generating a seeded array and then choosing random numbers corresponding to the position in the array and decreasing the size of the array, investigate established algorithms

Implementation:

* Basic C# console application with a goal of not spending more than a couple of hours of development time
* Investigation of this type of concept led me to: <http://stackoverflow.com/questions/962802/is-it-correct-to-use-javascript-array-sort-method-for-shuffling/962829#962829>
* Further investigation led me to <http://en.wikipedia.org/wiki/Fisher-Yates_shuffle>, and a C# implementation of such an algorithm <http://www.dotnetperls.com/fisher-yates-shuffle>
* Notes on this: While I feel I could have developed something from scratch that would have accomplished this task, it however would be foolish to try and develop my own algorithm when there are established methodologies that have been extensively tested. I don’t presume to take credit for this algorithm and am citing my sources above, but at the same time Pandell as a customer would only care about quality output. My research suggested that this is highly reliable as far as the randomness of the results, and is also an in place algorithm not requiring any extra memory. There were other faster methods, which were not in place, but to meet the requirements of this project it seemed quite reasonable.
* Developing the menu system was fairly straight forward, and emphasis was placed on checking user input for commands out of the expected options, out of range of values, or opposite to the goals of the program
* Allowing for full int32 arrays developed memory exceptions, while on my machine I was able to produce a list over 50 million numbers via the function I implemented I am unsure of the memory capabilities of any machine this might be ran on so I will cut it down to a max of 1 million (which far exceeds the project requirements). This was honestly the first time I had ran into this exception so it also prompted me to do some background research on what memory would be accessible to this program.
* A system was developed to allow for the user to receive notification as the Random Number Text file was being written, however even with the million numbers top end, writing the file was almost instantaneous so I commented it out, it might be useful if the maximum number increased.

Verification:

* I tested the system trying to break things, and enter commands that were out of bounds, negative values, etc
* Corrected any input verification errors, and tried to make exception handling more accurate in regards to the types of exceptions generated, and then tailor that feedback to the user.

Review and Maintenance:

* For the moment this is a “one of” system, and short of feedback on this project it will stand without revision. I did try and build it as a bit beyond spec to add value to the project and have it stand on its own (and hopefully stand out!).
* I considered breaking the project up into a couple supporting classes or possibly incorporating generic support but that seems to far beyond scope (again not trying to send someone to the moon with this project). I do realize that this is an evaluation project, but at the same time I am also placing a value around not over engineering this project.
* It would be also interesting to examine how to make the set of numbers higher, and I could see something with database interaction being very large, but also magnitudes slower

Hopefully as I stepped through those sections as I developed the solution to this project my notes will give you some insight into my problem solving methods and the consideration I took in approaching this task.