Homework 2 Grade Sheet

Name : Kyle James

|  |  |  |  |
| --- | --- | --- | --- |
| **Part** | **Criteria** | **Max** | **Earned** |
| **Code** | * CustomList class * Class is generic * Class has only two fields * Properties * Count * Capacity * Indexer * Constructors * Default * Parameterized * Methods * Add * IndexOf * Contains * RemoveAt * Remove * Insert * Clear * Print * Main Method * Gets user input and stores in the list * Special commands * done * print * indexOf * contains * removeAt * remove * insert * clear * get * set * CustomList<int> tested and working | **3**  **4**  **2**  **2**  **5**  **3**  **3**  **5**  **5**  **5**  **6**  **6**  **7**  **3**  **2**  **3**  **2**  **2**  **2**  **2**  **2**  **2**  **2**  **2**  **2**  **2**  **6** | 3  4  2  2  5  3  3  5  5  5  6  6  6  3  2  3  2  2  2  2  2  2  2  2  2  2  6 |
| **General** | * Follows C# coding standards for commenting, indentation and naming | **10** | 10 |
|  | Total | **100** | 99 |

***Comments:***

Overall, quite nicely done!

There's a couple things worth mentioning:

1) Insert() should simply call Add() if the provided index is >= count, instead of doing nothing (only for when index < 0).

2) You overall have the right idea with your Insert() implementation, but it could be simplified a little bit (no points lost in regards to this though). Here's an example potential implementation that showcases a few simplifications:



3) With simplification in mind, we can also take a look at Add() - if you find yourself repeating code in two separate blocks within the same method (or really anywhere for that matter), always take a moment to explore seeing if you can find a way to call it once, leading to more maintainable code. Here's an example potential simplified implementation for Add():



In general, I like to keep the "D.R.Y." principle in mind (Don't Repeat Yourself!)

4) RemoveAt(), while working, could be greatly simplified as well. We don't need to create a new array as we know we're only removing data (so capacity can stay the same), and we simply just want to shift data backwards, thus overwriting the data at the index, and reduce count (thus removing access to the [previously accessible] data at [what is now] count + 1). Here's what a simpler RemoveAt() might look like:



5) Clear doesn't need to reinitialize an array, simply just setting count to 0 will ensure that any new data added will overwrite any existing data already there.