**What I’ve learned can only be applied to the exercises (not all of them is necessary but can’t use anything beyond what should have been learned below:**

**Classes & Methods -**

* Declare, assign, and use variables to hold and change values in your program.
* Use basic math operations.
* Take user input and display messages to the user
* Declare, initialize, use, and traverse arrays
* Use conditional statements to control which parts of a program run
* Ability to combine multiple conditional checks into a single line
  + ex.  if (value > 0 && value < 10) …
* Use loops to repeat sections of your program and keep the program running until the user chooses to exit.
* Reuse sections of code by implementing methods

**Accessing Classes -**

* Overload methods and call the different versions
* Declare a class
* Add fields to a class
* Add methods to a class
* Define constructors for a class
* Instantiate class objects and use them
* Declare static fields in a class and use them
  + Know when to not use them (most of the time).

**Inheritance & Polymorphism -**

* Understanding of Object Oriented Programming concepts
* Knowledge of Encapsulation and ability to apply its concepts
* Declare and use C# Properties
  + Know when to use properties vs methods
* Create children classes
  + Take advantage of what your parent classes do for their children

**Interfaces & Abstract Classes –**

* Understanding of Abstract classes and how to use them
  + Know how to use Override with an Abstract class
* Understanding of Interfaces and how to use them
  + Know how to implement an Interface
* Know the difference between Implementation vs Inheritance

**Debugging –**

* Trouble shoot an application that will not compile
* Debug an applicaiton that shows unexpected behavior
* Set/Remove break points
* Step through your code a line at a time
* Use the watch window to monitor the values of variables you select

**Generics and Collections –**

* Understand how to store data in related, organized schemes.
* Understand how to use templated container types.
* Be able to identify which collection should be used to store sets of data.
* Be able to create and manage data in a generic list.
* Be able to create and manage data in a dictionary.

**File I/O –**

* Understand how to store data in a persistent file.
* Be able to read and write data to a file using streams.
* Understand the JSON scheme.
* Be able to read, translate, and write data to / from the JSON format.

**Sorts –**

* Understand the purpose of organizing data.
* Understand the criteria needed to sort data.
* Be able to organize data using the Bubble-sort algorithm.
* Be able to organize data using the Insertion-sort algorithm.

**Binary Search and Recursion –**

* Understand how sorting data relates to efficient searching through data.
* Understand how function recursion can mimic looping behavior.
* Be able to perform a Binary-search using loop flow.
* Be able to perform a Binary-search using recursion.