|  |  |  |
| --- | --- | --- |
| Concept | Score | Notes on how you may improve your score or on your score |
| Classes and Objects | 5 |  |
| Inheritance | 3 |  |
| Virtual Functions | 2.5 | Will need to re-read chapter again and play around with a simple test program   * Mostly confused by the way you call base class and derived class functions using pointers (don’t understand why you need that, etc.) |
| Command Line Arguments | 5 |  |
| Template Function | 3 |  |
| Template Class | 3 |  |
| STL List | 3 |  |
| STL Queue | 2 | I’ve read the chapter dealing with Queues, Stacks etc – I understand the concept  And understand the way they’re implemented, I just haven’t used them in a program  Or played around with it in a simple test program enough to be able to confidently say I could apply them or teach another person how to implement it. I could probably teach a person the concept behind it though, but not the C++ code. |
| STL Stack | 2 |
| Write Own List | 2 |
| Write Own Queue | 2 |
| Write Own Stack | 2 |
| Abstract data types | 2 | I read the chapter multiple times, but still don’t completely understand the idea or what it was getting at – I feel like its actually more simplistic of a concept than I’m thinking. I will need to re-read it again or find some other sources that can help supplement the text. |

* What part of the program are you most proud of and why?
  + The gameboard class is probably the most advanced portion of the code, though its not really that complex. Getting the gameboard vector to pass as one whole vector was one of the more challenging aspects of the class, but is also ironically one of the smallest sections of that code block.
* What challenges did you face and how did you solve them?
  + It took me a long time to understand what the human player class would do honestly. Logically a player would: 1.) place their own ships and 2.) attack positions.
    - But placing your own ships, is done for you as part of the file input process, so there’s really no need for that function as of right now (perhaps its needed later in part 2 or part 3)
    - And the second function attacking positions can’t be implemented in the human class really since it requires user input and output which shouldn’t be a part of a class.
    - If you do the input and output outside the class, you could send that information to an AttackPosition function of the class, but that’s needless as it would then just send that information to another function: my code to check a position is part of the opponent’s class, and more specifically part of the gameboard object that the opponent class holds.