|  |  |  |
| --- | --- | --- |
| Concept | Score | Notes on how you may improve your score or on your score |
| Classes and Objects | 5 |  |
| Inheritance | 3 |  |
| Virtual Functions | 3 | Learned a bit more about their function and use, and the usage of pure virtual functions when building the player class |
| Command Line Arguments | 5 |  |
| Template Function | 3 |  |
| Template Class | 3 |  |
| STL List | 4 | STL list, queue and stack I am comfortable with the concept behind them and the specific way in which they are implemented and their functions |
| STL Queue | 4 |  |
| STL Stack | 4 |  |
| Write Own List | 2 | I understand the way in which the lists, queues and stacks work outside the STL libraries – they way they function, the way in which each element links to each other, the concept of the head and tail, etc. but I am still unclear in the way in which they are built – that is the specific code used in their creation. |
| Write Own Queue | 2 |  |
| Write Own Stack | 2 |  |
| Abstract data types | 3 |  |

* What part of the program are you most proud of and why?
  + The main portion of the new programming for Project 2 Part B is in the computer class, specifically the attackPosition function. It holds the programing that allows the computer opponent to make simple “decisions” relating to it’s choices of where or what to attack. The rest of the changes to the code is mostly older concepts or easier concepts, like the arguments section of the code.
* What challenges did you face and how did you solve them?
  + The main challenge was probably coming back to the code after a while and attempting to start adding components or building sections of the code to tackle new problems. Its easy to start adding some simple components at first like the section requiring arguments, but then gets much more complicated trying to remember how you structured the classes and how they should interact.
  + So the solution is two fold, my comments are perfectly fine and helped me out quite a bit, but my function names or variable names could have been much more specific, so changing them or developing a visual map or document to explain things to future self would have been good. And the second part of the solution is just sitting down and re-reading the program and stepping through it logically before jumping into programming the new components. (example of confusing identifiers or concepts: calling the gameboard class boardgame at first, and visa versa – and- calling the functions relating to player class object’s board a PersonalMap, which is in turn retrieving information from its gameboard object’s gameboard vector – essentially re-deciphering the way I structured things)
  + The other problem was just the way in which I implemented the attackPosition function for the computer class, specifically how to write the functions’ flow. I had to split it into two logical sections one prior to checking the result of the attack coordinates, and one afterwards. The one before checks to see if the last attack was a Miss and also if the Queue exists – if it doesn’t it randomly guesses. If that doesn’t execute it checks to see if the Queue does exist, then since we push all values to the queue before checking it has to recheck the positions [would have been easier to check before pushing to the Queue, but may cause some other unseen problem].
    - The second part of the code consists follows the subsequent testing of the position for a hit/miss -- and if it is a Hit, it immediately creates the Queue