Project 3 Report

1. I used several data structures to design the battleship game. My board implementation uses a two-dimensional character array of size Maxcols, MaxRows to store all of the data for each player’s set-up. Shots are stored in the array as characters according to whether they hit or missed. The rest of my data structures were all vectors. In my Game implementation, I used a vector to store pointers to all of the ships added to the game this allows an infinite number to be added, or just the five standard ones. In my Player implementations for the mediocre and goodplayer’s, I store the locations of previous shots in the form of an integer by multiplying the row by 10 and adding the column (so (3,5) would be stored as 35) this allows me to use find to search the vector to see if a proposed attack, for example, has already been executed.
2. My goodplayer places its ships the same way every time, with good spacing such that when a mediocre player enters state 2 of shot selection after hitting one ship, none of those shots will overlap with and hit another of good player’s ships. To recommend an attack, it uses a similar algorithm as the mediocre player only with some improvements. It attacks randomly in a checkerboard pattern, rather than attacking any square randomly. When it hits, it does the same thing as mediocre player, only if the aircraft carrier has been eliminated, it attacks in a cross pattern that ranges only 3 squares out from the hit point. Once all of the checkerboard spots are covered, each ship must be hit so it attacks random spots around previously hit spots.
3. Goodplayer’s recommendAttack:

If we just sunk the aircraft carrier, Longest ship is put at 4,

This affects the state 2 targeting, reducing range

If a shot was hit, push it onto the hits vector

If we were in state 1 or 3 and hit a shot, we now go to the targeting state and The hitpoint is set to the last cell attacked

If a ship was destroyed go back to state 1

While in state 1

If we completed all random checkerboard patterned attacks, go to state 3

Otherwise, keep trying random points we haven’t attacked yet

i.e. ones that are not in the aps vector

until we hit something

then wed go to state 2,

or the whole board is checkered then we go to state 3

in state 2,

we attack the (at most) 18 positions around a hit,

we create an array to store these and attack them randomly. Once we finish attacking all of those or sink the ship,

we go back to state 1/3

in state 3

once weve attacked every other square,

we know that we must have hit everyship and therefore

we look at previously hit points

and attack points randomly,

adjacent to those points.