Project 1

Battleships

Final

CIS 17c

Thomas Gustafson

10/21/19

Introduction

Battleships has two players face off against each other they each have their own board that is 10 x 10. They will place five ships of length five, four, three, three, and two. The players then will alternate calling out positions to attack till one player has no ships remaining.

Summary

Project size:750+ lines

I expanded upon my midterm project adding hashing and binary trees. I had done some extra research for my midterm project and used merge sort, so I had less work to do for the final project. I used binary trees instead of a priority queue to display the leaderboard. As for hashing I used that as a small demo. I used it to store the winner’s names in an array of linked lists and count how many unique winners there had been.

*Main*

*Prompt user for which game mode*

*If leaderboard mode*

*Create binary tree for leaderboard*

*Do*

*Create Battleships object*

*Prompt user if they would like to continue once game is over*

*While user says continue*

*Output leaderboard*

*Else if timed mode*

*Create battleships object*

*End*

*Board*

*Constructor*

*Mark all map spots with the string empty*

*SetString*

*Make the string go into the matching key*

*GetOccup*

*Return true or false on whether board position is occupied*

*PrintBoard*

*Print all elements for the board using an iterator and the keys to determine when*

*A ne line should be created*

*Player*

*Constructor*

*Set moves to 0*

*Create list with 5 ships*

*Prompt user name*

*Set status to true*

*Set pieces to 17*

*For loop for input of each ship*

*Set spaces of ships to occupied*

*GetStatus*

*Return active status*

*setStatus*

*Set player active status*

*GetOccu*

*Return if boardspace is occupied*

*SetBoard*

*If hit decrement pieces*

*Set board element*

*PrintBoard*

*Call printboard from board*

*AddMove*

*Increment move*

*GetMove*

*Retunrn move*

*Battleships*

*Constructor*

*Create and array of 2 player objecs*

*Set game status to true*

*PlayGame*

*While game equals true*

*Print board*

*Prompt User for attack location*

*Call attack*

*If else to determine what name to output and player to return*

*Attack*

*If getOccu returns true*

*setBoard hit*

*output hit*

*Else if getOccu returns false*

*setBoard miss*

*output miss*

*TimedMode*

*Create stack to track playertime*

*While game equals true*

*Print board*

*Prompt User for attack location*

*Call attack*

*Push player time for turn to stack*

*Use stack to determine total time for each player*

*Calculate score by multiplying pieces remaining and time*

*If else to determine what name to output and player to return*

*BinaryTree*

*Insert*

*Check if value is greater than or less than root and pass down till meeting null*

*Call balance*

*Balance*

*Check if height of left and right tree have a difference of 2 or more*

*Check branch balance*

*Rotate depending on length*