Jordan Golan

UID: 405328335

1. The main obstacle that I faced while completing this project was syntax errors while using and implementing classes. For example, after basically completing all the code, a compile error appeared. This is because when I tried to implement the winningBid function from the Auction class. I wrote “Action::a.winningBid()”, however then “Auction::” was not needed and caused the error. This ended up getting rid of eight compile errors throughout my code that said I was redeclaring the Auction class. Another obstacle that I encountered during this project was writing the bid function. I repeatedly misinterpreted the directions and the how the function should work. For example, I believed that a bid should only count if it was greater than the minimum, however as long as it is greater than the previous bid it counts. I figured this out through CodeBoard testing.
2. Below is a list of asserts and a reason for each

Auction auction1("Television", 100.00);

EBayBucks bucks1;

// testing if the get description function works

assert(auction1.getDescription () == "Television");

// number of bids at beginning is 0

assert(auction1.numberOfBids() == 0);

auction1.openAuction();

// test no negative bids

assert(auction1.bid(-10.00)==false);

// test a bid under minimum price is good

assert(auction1.bid(20.00)==true);

// test a bid lower than previous bid is false

assert(auction1.bid(10.00)==false);

// test a bid greater than previous bid is true

assert(auction1.bid(30.00)==true);

// test wasSuccessful is false if auction is still open

assert(auction1.wasSuccessful()== false);

// test winningBid is -1 when Auction is open

assert(std::to\_string(auction1.winningBid()) == "-1.000000");

auction1.closeAuction();

// test wasSuccessful is false if bid is less than min

assert(auction1.wasSuccessful()== false);

// test winnningBid is -1 when wasSuccessful is false

assert(std::to\_string(auction1.winningBid()) == "-1.000000");

auction1.openAuction();

// giving a bid greater than min

assert(auction1.bid(200.00)==true);

auction1.closeAuction();

// test wasSuccessful is true is bid is greater than min

assert(auction1.wasSuccessful()==true);

// test numberOfBids

assert(auction1.numberOfBids()==3);

// test winningBid when auction is successful

assert(std::to\_string(auction1.winningBid()) == "200.000000");

bucks1.addAuction(auction1);

// test earnings after one auction

assert(std::to\_string(bucks1.earnings()) == "2.000000");

Auction auction2("Table", 50.00);

// create a second auction

auction2.openAuction();

assert(auction2.bid(10.00)==true);

// create an unsuccessful auction

auction2.closeAuction();

bucks1.addAuction(auction2);

// unsuccessul auctions don’t add to earnings

assert(std::to\_string(bucks1.earnings()) == "2.000000");

//create a third successful auction

Auction auction3("xBox", 300.00);

auction3.openAuction();

assert(auction3.bid(400.00)==true);

auction3.closeAuction();

bucks1.addAuction(auction3);

// earning should be cumulative

assert(std::to\_string(bucks1.earnings()) == "6.000000");

// issue certificate should be cumulative

assert(std::to\_string(bucks1.issueCertificate()) == "6.000000");

// after issue certifiate is called earnedAward should be 0

assert(std::to\_string(bucks1.earnings()) == "0.000000");