Kevin Wang

UID: 205209507

Project 5 Report

Notable obstacles:

* I was experiencing an error that stated "unresolved external symbol" which I found was a linker error. It took me awhile to figure out to how solve the issue even after I googled the error statement. I ended up fixing it after looking at my solution explorer and saw that one of my .cpp weren’t listed in the source files despite it being open within my solution. I added the existing file to the source file directory and it resolved my issue.
* Within MegaMillionsLottery::WinningPossibility, I initially used a switch statement to return the enumerated winning possibility based off of the counter variables I put in place to see how many balls matched. The code compiled on visual studios but when put into codeboard and when I tried to compile in g++, I would get an error that sated the function would reach the end of the control without a return value. I had to then switch the switch statement to a bunch of if statements and that ended up resolving the issue.
* Some of the syntax was hard to figure out such as the parentheses after getBall1(); and within the constructor for MegaMillionsLottery where we needed to supply randomly generated values. It took me awhile to figure out how to correctly write out RandomNumber r1(1,75) and then the following step that assigned it to the private data members to make mBall1 = r1.random();

Test Data:

MegaMillionsTicket ticket(1, 2, 3, 4, 5, 6);

assert(ticket.getBall1() == 1);

assert(ticket.getBall2() == 2);

assert(ticket.getBall3() == 3);

assert(ticket.getBall4() == 4);

assert(ticket.getBall5() == 5);

assert(ticket.getMegaBall() == 6);

* Used to test that the MegaMillionsTicket constructor and accessors correctly returned the right values.

MegaMillionsLottery lottery(1, 2, 3, 4, 5, 6);

assert(lottery.getBall1() == 1);

assert(lottery.getBall2() == 2);

assert(lottery.getBall3() == 3);

assert(lottery.getBall4() == 4);

assert(lottery.getBall5() == 5);

assert(lottery.getMegaBall() == 6);

* Used to test MegaMillionsLottery constructors and acessors returned the correct values when the values are supplied into it

assert(lottery.checkTicket(ticket) == MegaMillionsLottery::WinningPossibility::FIVEPLUSMEGABALL);

assert(lottery.whatHappened(ticket) == "You matched 5 balls plus the megaball!");

ticket = MegaMillionsTicket(1, 2, 3, 4, 5, 12);

assert(lottery.checkTicket(ticket) == MegaMillionsLottery::WinningPossibility::FIVE);

assert(lottery.whatHappened(ticket) == "You matched 5 balls!");

* Check that the correct WinningPossibility returned the correct enumerated winning possibility
* Also checks whatHappened has the right message

MegaMillionsTicket quickPickTicket(1, 2, 3, 4, 5, 6);

for (int i = 0; i < 20; i++)

{

quickPickTicket = lottery.quickPick();

// all the ball numbers need to be different...

assert(quickPickTicket.getBall1() != quickPickTicket.getBall2() &&

quickPickTicket.getBall1() != quickPickTicket.getBall3() &&

quickPickTicket.getBall1() != quickPickTicket.getBall4() &&

quickPickTicket.getBall1() != quickPickTicket.getBall5() &&

quickPickTicket.getBall2() != quickPickTicket.getBall3() &&

quickPickTicket.getBall2() != quickPickTicket.getBall4() &&

quickPickTicket.getBall2() != quickPickTicket.getBall5() &&

quickPickTicket.getBall3() != quickPickTicket.getBall4() &&

quickPickTicket.getBall3() != quickPickTicket.getBall5() &&

quickPickTicket.getBall4() != quickPickTicket.getBall5());

}

* Checks that the quickPickTicket function correctly gets a ticket with all different values
* Also checks that the MegaMillionsLottery () constructor with no argument correctly randomly chooses ball values and make sure they are all different.