Artiom Arutiunov

Student ID: 504597668

Professor Stahl

2/24/16

Project 5 Report

a) There were several notable obstacles that I had to overcome while working on this project. First and foremost, I found it challenging working with classes and figuring out proper formatting. After figuring out the difference between .h and .cpp files, as well as properly implementing functions within the .cpp file, I began to actually write the code for the classes, which was not an easy task. Next, I ran into trouble when writing the code for the default constructor of the PowerballLottery class. What I eventually realized is that I didn’t create six separate objects of the RandomNumber class when creating the five exclusive lottery numbers on top of the sixth powerball. Before, my single object would become redefined every time I called the random function. I additionally had some trouble understanding the directories of the type of functions defined with the .cpp files. For example, it wasn’t obvious to me that when returning one of the enumeration types defined in the class, I had to include where it was stored followed by the two colons (::). Lastly, I encountered a temporary roadblock when working with an object of a different class within another class. Specifically, when writing the code for the quickPick function, I didn’t realize right away that I first had to create an object of class PowerballLottery in order to give the object of PowerballTicket six random digits as its parameter.

b) I split my test cases into two separate blocks: the cases that Howard provided in the spec and cases that I came up with myself.

Howard’s Test Cases

1) Testing accessor methods for PowerballTicket work

PowerballTicket 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.getPowerball() == 6);

🡪 The compiler agreed with all of these assertions, which means the getBall functions of class PowerballTicket work properly.

2) Testing accessor methods for PowerballLottery work

PowerballLottery 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.getPowerball() == 6);

🡪 The compiler agreed with all of these assertions, which means the getBall functions of class PowerballLottery work properly.

3) Testing PowerballLottery quickPick function

PowerballTicket 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());

}

🡪 The compiler agreed with all of these assertions, which means the quickPick function (on top of getBall) works correctly by not allowing duplicates among the first five numbers of the lottery ticket.

My own test cases

1) Testing *predefined* PowerballLottery checkTicket & printWhatHappened functions

PowerballLottery lotteryA1(1, 2, 3, 4, 5, 6);

PowerballTicket ticketA1(1, 2, 3, 4, 5, 6);

assert (lotteryA1.checkTicket(ticketA1) == PowerballLottery::WinningPossibility::FIVEPLUSPOWERBALL);

lotteryA1.printWhatHappened(ticketA1);

PowerballLottery lotteryA2(1, 2, 3, 4, 5, 7);

PowerballTicket ticketA2(1, 2, 3, 4, 5, 6);

assert (lotteryA2.checkTicket(ticketA2) == PowerballLottery::WinningPossibility::FIVE);

lotteryA2.printWhatHappened(ticketA2);

PowerballLottery lotteryA3(1, 2, 3, 6, 7, 6);

PowerballTicket ticketA3(1, 2, 3, 4, 5, 6);

assert (lotteryA3.checkTicket(ticketA3) == PowerballLottery::WinningPossibility::THREEPLUSPOWERBALL);

lotteryA3.printWhatHappened(ticketA3);

PowerballLottery lotteryA4(1, 2, 3, 6, 7, 6);

PowerballTicket ticketA4(1, 2, 3, 4, 5, 61);

assert (lotteryA4.checkTicket(ticketA4) == PowerballLottery::WinningPossibility::THREE);

lotteryA4.printWhatHappened(ticketA4);

PowerballLottery lotteryA5(1, 2, 3, 6, 7, 61);

PowerballTicket ticketA5(1, 2, 9, 4, 5, 61);

assert (lotteryA5.checkTicket(ticketA5) == PowerballLottery::WinningPossibility::TWOPLUSPOWERBALL);

lotteryA5.printWhatHappened(ticketA5);

PowerballLottery lotteryA6(1, 2, 3, 6, 7, 61);

PowerballTicket ticketA6(1, 2, 9, 4, 5, 62);

assert (lotteryA6.checkTicket(ticketA6) == PowerballLottery::WinningPossibility::NOTWINNING);

lotteryA6.printWhatHappened(ticketA6);

PowerballLottery lotteryA7(1, 2, 3, 6, 7, 61);

PowerballTicket ticketA7(1, 11, 9, 4, 5, 62);

assert (lotteryA7.checkTicket(ticketA7) == PowerballLottery::WinningPossibility::NOTWINNING);

lotteryA7.printWhatHappened(ticketA7);

PowerballLottery lotteryA10(1, 2, 3, 6, 7, 61);

PowerballTicket ticketA10(0, 11, 9, 4, 5, 62);

assert (lotteryA10.checkTicket(ticketA10) == PowerballLottery::WinningPossibility::NOTWINNING);

lotteryA7.printWhatHappened(ticketA10);

PowerballLottery lotteryA8(1, 2, 3, 4, 7, 6);

PowerballTicket ticketA8(1, 2, 3, 4, 5, 6);

assert (lotteryA8.checkTicket(ticketA8) == PowerballLottery::WinningPossibility::FOURPLUSPOWERBALL);

lotteryA8.printWhatHappened(ticketA8);

PowerballLottery lotteryA9(1, 2, 3, 4, 7, 6);

PowerballTicket ticketA9(1, 2, 3, 4, 5, 61);

assert (lotteryA9.checkTicket(ticketA9) == PowerballLottery::WinningPossibility::FOUR);

lotteryA9.printWhatHappened(ticketA9);

🡪 The compiler agreed with all of these assertions and printed out the corresponding messages to them. This also means that the compiler recognized the enumeration in the PowerballLottery class.

2) Ensuring that default constructor of PowerballLottery does not allow duplicates in the first five numbers.

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

{

PowerballLottery ticketswag;

cerr << ticketswag.getBall1() << endl;

cerr << ticketswag.getBall2() << endl;

cerr << ticketswag.getBall3() << endl;

cerr << ticketswag.getBall4() << endl;

cerr << ticketswag.getBall5() << endl;

cerr << ticketswag.getPowerball() << endl;

cerr << “---“ << endl;

}

🡪 I did not see any duplicates among the five lottery numbers, which means that the default constructor of PowerballLottery is working correctly.

3) Ensuring that PowerballLottery checkTicket & printWhatHappened functions work properly with *random* numbers.

for (int i=0; i<30; i++)

{

PowerballLottery lotteryB1(1, 2, 3, 4, 5, 6);

PowerballTicket ticketB1(1, 2, 3, 4, 5, 6);

ticketB1 = lotteryB1.quickPick();

lotteryB1.printWhatHappened(ticketB1);

PowerballLottery lotteryB2(1, 2, 3, 4, 5, 6);

PowerballTicket ticketB2(1, 2, 3, 4, 5, 6);

ticketB2 = lotteryB2.quickPick();

lotteryB2.printWhatHappened(ticketB2);

PowerballLottery lotteryB3(1, 2, 3, 4, 5, 6);

PowerballTicket ticketB3(1, 2, 3, 4, 5, 6);

ticketB3 = lotteryB3.quickPick();

lotteryB3.printWhatHappened(ticketB3);

}

🡪 The 30 iterations demonstrated that the functions in fact work with random numbers, and there is variety within the results (most of them were NOTWINNING).

4) Ensuring that PowerballLottery random numbers are within the valid value parameters

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

{

PowerballLottery check;

assert( check.getPowerball() >= 1 && check.getPowerball() <= 26);

assert( check.getBall1() >= 1 && check.getBall1() <= 69);

assert( check.getBall2() >= 1 && check.getBall2() <= 69);

assert( check.getBall3() >= 1 && check.getBall3() <= 69);

assert( check.getBall4() >= 1 && check.getBall4() <= 69);

assert( checkgetBall5() >= 1 && check.getBall5() <= 69);

}

🡪 Compiler agreed with all of these assertions, which means that the PowerballLottery numbers are all within the correct ranges.