**PROJECT 5 REPORT**

1. Some of the more notable obstacles that I overcame while coding this Project were first of all, the idea of classes and the implementation .cpp file that went along with classes. I wasn’t sure exactly how to link the .cpp file with the .h file, and it took me about an hour to figure out that C++ would do it automatically as long as I included the header in the .cpp file. This was important to note, but it also took up a lot of my time. Another major obstacle that I went through while coding was trying to put two headers together. I was trying to implement PowerballTicket.h into PowerballLottery.h and it would give me a lot of linker compiler errors, and I was very confused as to why this was happening. I couldn’t really figure it out and had to look around for other sources. It turns out that it is an error to link two header files together like that. I was trying to put in the class PowerballTicket into the header PowerballLottery, and this requires a forward declaration of:

class PowerballTicket;

This line of code took me a good hour to figure out because I wasn’t sure why I was getting these compiler errors since I tried to include the header within another header. After I declared the forward declaration, however, everything went well.

Another error of interest that occurred was when I was writing the if statement trees in order to assign the return value for the checkTicket member function. For some reason, I would keep on getting “POWERBALL” returns even though some of the other numbers were matching. I figured out that I had written

if (array1[5] == array2[5])

{

return POWERBALL;

}

else if (matches == 1 && array1[5] == array2[5])

This was an error because if the powerball number matched, then the if statement tree would break there and it wouldn’t go into the “else” statements, just by how it was designed. I ended up solving this by writing “&& matches == 0” for this statement, and “&& array1[5] != array2[5]” for the other cases where only the numbers were matched and not the powerball number.

Other than that, the only other major obstacles that I had to go through were very simple mistakes, such as accidentally writing “getBall4()” while assigning a number of mBall5, which was extremely confusing for me because I kept on getting an extra counter in that position. It was actually very lucky that I managed to catch this error.

Another small thing worth mentioning was that I struggled slightly with putting all the elements of the PowerballTicket and PowerballLottery classes into arrays. I wasn’t sure how to do it until I remembered that we had written the getBall() function for this very reason.

If I encountered any small difficulties along the way, I have also written comments detailing my thought process on my actual code.

1. The test cases that I used were as follows:

I used the test code that was given in the actual problem statement, which just assigned a couple of numbers to a class PowerballTicket called ticket, and then asserted that each number was correct.

Then, I put in random numbers into “chance” in order to test the same thing.

The last part of the code ran “quickPick” to see if the numbers were all different.

I, however, ensured that my function quickPick worked (because between 1-69, there could have been a possibility of random number generation where the numbers were all different, although the function could produce the same number). As such, I actually reduced the RandomNumber function from 1-69 to 1-6, which gave only 6 possible numbers, meaning that with only a pool of 6 numbers to choose from, my code could be tested to see if it really worked. It passed this test.

The test cases above were very specific, and the test cases that I list below are meant to test the checkTicket function and the printWhatHappened function.

**POWERBALL**

In order to test if my code could work when just the powerball matched, I used a ticket of [1, 2, 3, 4, 5, 6] and a winning combination of [6, 7, 8, 9, 10, 6]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**ONE PLUS POWERBALL**

In order to test if my code could work when the powerball and one number matched, I used a ticket of [4, 5, 6, 7, 8, 10] and a winning combination of [11, 12, 13, 14, 4, 10]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. I also switched the numbers around so that I could test if the order of numbers being compared would matter or not (it shouldn’t). Test succeeded.

**TWO PLUS POWERBALL**

In order to test if my code could work when the powerball and two numbers matched, I used a ticket of [4, 5, 6, 7, 8, 11] and a winning combination of [6, 7, 11, 12, 13, 11]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. I also switched the numbers around so that I could test if the order of numbers being compared would matter or not (it shouldn’t). Test succeeded.

**THREE**

In order to test if my code could work when three numbers matched, I used a ticket of [1, 2, 3, 9, 10, 14] and a winning combination of [1, 2, 3, 4, 5, 7]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**THREE PLUS POWERBALL**

In order to test if my code could work when three numbers and the powerball matched, I used a ticket of [1, 2, 3, 9, 10, 7] and a winning combination of [1, 2, 4, 3, 5, 7]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**FOUR**

In order to test if my code could work when four numbers matched, I used a ticket of [1, 2, 3, 4, 9, 10] and a winning combination of [1, 2, 3, 4, 5, 6]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**FOUR PLUS POWERBALL**

In order to test if my code could work when four numbers plus the powerball matched, I used a ticket of [1, 2, 3, 4, 9, 6] and a winning combination of [1, 2, 3, 4, 5, 6]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**FIVE**

In order to test if my code could work when five numbers matched, I used a ticket of [21, 22, 23, 24, 25, 26] and a winning combination of [21, 22, 23, 24, 25, 27]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**FIVE PLUS POWERBALL**

In order to test if my code could work when five numbers plus the powerball matched, I used a ticket of [31, 32, 33, 34, 35, 36] and a winning combination of [31, 32, 33, 34, 35, 36]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**NOTWINNING**

In order to test if my code could work when nothing matched (which is usually the case in a real lottery), I used a ticket of [4, 56, 34, 50, 28, 4] and a winning combination of [5, 49, 10, 23, 60, 9]. I then printed out what printWhatHappened would say and compared that with what was supposed to happen. Test succeeded.

**ACTUAL LOTTERY (RANDOM NUMBERS)**

In order to test if my random functions would work, I simulated an actual lottery by calling PowerballLottery() in order to test if the random number generator for that would work. I then both used a quickPick ticket generator and input my own values. Unfortunately, I didn’t win anything, but my code worked and the test succeeded.

The code that I used to test can be seen below:

#include <iostream>

#include <string>

#include <cassert>

#include "PowerballTicket.h"

#include "PowerballLottery.h"

int main()

{

//test code

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);

PowerballTicket chance(1, 45, 27, 9, 32, 22);

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

assert(chance.getBall2() == 45);

assert(chance.getBall3() == 27);

assert(chance.getBall4() == 9);

assert(chance.getBall5() == 32);

assert(chance.getPowerball() == 22);

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);

PowerballTicket ticket2(2, 3, 4, 5, 6, 7);

PowerballLottery lottery2(2, 3, 4, 5, 6, 8);

assert(lottery.checkTicket(ticket) == 8);

assert(lottery2.checkTicket(ticket2) == 7);

assert(lottery.checkTicket(chance) == 9);

//lottery.printWhatHappened(ticket);

PowerballTicket tick1(1, 2, 3, 4, 5, 6);

PowerballLottery lott1(6, 7, 8, 9, 10, 6);

assert(lott1.checkTicket(tick1) == 0);

lott1.printWhatHappened(tick1);

PowerballTicket tick2(4, 5, 6, 7, 8, 10);

PowerballLottery lott2(11, 12, 13, 14, 4, 10);

assert(lott2.checkTicket(tick2) == 1);

lott2.printWhatHappened(tick2);

PowerballTicket tick3(4, 5, 6, 7, 8, 11);

PowerballLottery lott3(6, 7, 11, 12, 13, 11); // This will check if the order of numbers doesn't matter

assert(lott3.checkTicket(tick3) == 2);

lott3.printWhatHappened(tick3);

PowerballTicket tick4(1, 2, 3, 9, 10, 14);

PowerballLottery lott4(1, 2, 3, 4, 5, 7);

assert(lott4.checkTicket(tick4) == 3);

lott4.printWhatHappened(tick4);

PowerballTicket tick5(1, 2, 3, 9, 10, 7);

PowerballLottery lott5(1, 2, 4, 3, 5, 7);

assert(lott5.checkTicket(tick5) == 4);

lott5.printWhatHappened(tick5);

PowerballTicket tick6(1, 2, 3, 4, 9, 10);

PowerballLottery lott6(1, 2, 3, 4, 5, 6);

assert(lott6.checkTicket(tick6) == 5);

lott6.printWhatHappened(tick6);

PowerballTicket tick7(1, 2, 3, 4, 9, 6);

PowerballLottery lott7(1, 2, 3, 4, 5, 6);

assert(lott7.checkTicket(tick7) == 6);

lott7.printWhatHappened(tick7);

PowerballTicket tick8(21, 22, 23, 24, 25, 26);

PowerballLottery lott8(21, 22, 23, 24, 25, 27);

assert(lott8.checkTicket(tick8) == 7);

lott8.printWhatHappened(tick8);

PowerballTicket tick9(31, 32, 33, 34, 35, 36);

PowerballLottery lott9(31, 32, 33, 34, 35, 36);

assert(lott9.checkTicket(tick9) == 8);

lott9.printWhatHappened(tick9);

PowerballTicket tick10(4, 56, 34, 50, 28, 4);

PowerballLottery lott10(5, 49, 10, 23, 60, 9);

assert(lott10.checkTicket(tick10) == 9);

lott10.printWhatHappened(tick10);

// For fun, let's simulate an actual Lottery!

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

PowerballLottery actuallott = PowerballLottery(); // calling a random one!

PowerballTicket quickPickTick(5, 27, 42, 31, 18, 5); // my lucky numbers

actuallott.printWhatHappened(quickPickTick);

// I'm so unlucky...I haven't won anything, not even the Powerball

PowerballLottery test2(1, 2, 3, 4, 5, 6);

PowerballTicket quickPickTick2(1, 2, 3, 4, 5, 6);

quickPickTick2 = test2.quickPick();

actuallott.printWhatHappened(quickPickTick2);

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());

}

cout << "All tests succeeded!" << endl;

}