CS165 Assignment 2

Jason Dorweiler

1. Assignment Requirements

Part 2a is to start with the code from assignment one and use pow() cmath to calculate the max variable types.

The goal of 2b is to make a grade calculator. The calculator will take the number of assignments, score, and maximum points for each. It then will calculate the total, total possible, and a percentage for the grade.

For part 2c I will write a guess the number game using rand(). To play the game a user picks a number from 1 to 50. A total of 5 guesses is allowed. After each guess output if the guess is too high or too low.

In part 2d add a loop to the game to keep it running until the player wants it to stop.

In part 2e I will add something in the game that keeps track of the highest and lowest guess and output the bounds for the game based on the users past guesses.

In part 2f I will change the game to have the user enter a number and the computer will guess the answer.

2. Designing a solution

For part 2a I actually already implemented this on the previous assignment. I used pow() to calculate bit size of each variable type: for example

Max unsigned short = pow(2., sizeof(short)*8) =

For 2b:

Set up switch asking for the assignment score, max possible, or end calculation. Once the user picks to end the calculation it will print out the total score, total possible, and the student’s grade.

```
if ( enter_scores ) {

        Cin >> score, total possible

}

 If ( !enter_scores) {

        Cout << calculated grade, and total

}
```

For the guess the number game I can use cin to get the user input. I can set the random number using rand() % 50 + 1 to give it a range from 1 to 50. Probably add some error checking on the input like I did in the labs this week. Adding a while(replay) loop around the game will allow it to continue running for as long as the player wants. For the optional part of having the computer guess the game I will set it up to pick a random number somewhere in the range of the game. Then if it was incorrect then adjust the bounds of the game based on if the guess was too high or too low. Here's a rough plan that I came up with for the human player part of the game:

```
Int main(){

        while(replay) {

                // need to add some way to keep track of the lives left here??

                // while(lives)  check to see if the player still has lives

                number = rand()%50 + 1;

                cout << “Enter Guess”

                cin >> guess;

                if(guess > number){

                        cout << “Number is lower”;

                }

                if(guess < number){

                        cout << “Number is Higher”;

                }

        }

}
```

2. Implementing

See comments in the source files.

3. Testing and debugging

For the part of the assignment that asks to calculate the max/min variable types I included the limits.h library. I did this just so that I could use the macros to print out the actual answers just to double check. Then I would print out my calculated value along side the macro value.

I didn't have any trouble with the grade calculator. I end up going back and adding in an input error checking function to check that there is valid input from the user. This function assumes that you can't get a negative grade and check to see that the user didn't enter a non-integer.

For the guess the number game I was printing out the random number calculated by the computer just to check to see if it was doing what I though it should be when I picked a number too low/high. When a user enters a their numbers I check to see that it is within the original game range (1-10). If it's not then they are asked to enter it again. On the start screen I check to see if the value entered is 1 or 2, and no cin errors.

4. Reflection

Everything in this assignment was easy to understand and I didn't run into any big problems completing it. One thing I did was for the guess the number game. I had originally started writing it as I had planned in the designing part of this lab. The problem was that it was all in one big block of code in main(). The game was a bit more complicated that I though it was going to be and I didn't get too far before this became my code became hard to follow. I decided to scrap that plan and break down the game into specific functions that I could reuse for other parts of the game. I made functions for the start scree, clearing the scree, and for each type of player (human or comp). This worked out really well and made the code much easier to put together without any errors.