**Project 1 Design Document**

**Methods:** For storing information on the jugs, I used two sets of integers. A jug1 and jug2 that stores the information on the total amount that the jugs can hold, and then currJug1 and currJug2 that hold the current status of the jugs, and how much water is currently being held in them. The total amount of water needed is just stored in an integer. There is an integer counter that keeps up with how many steps the program was able to get the answer in. The last big structure is an array of strings that the outputs are saved to. This compares the saved string with previous saved strings, to cut the program off if it has begun to repeat itself.

**Functions:** From main, we go directly into setup() which takes and checks user input. It then prints out the start of the output chain, as seen in the project instructions. We pass the user inputted amounts into the next function operations(). This function is by far the largest. It starts by checking to see if the amount of water needed is met by either jug1 or jug2 by themselves. It then checks to see if jug1 is greater than jug2, if it is, the program prints a message saying it is rearranging the input to work nicer with the program, and subsequently clears, and re-outputs a fixed introductory set of lines. The next check is to see if the combined total of the two jugs is equal to the amount needed. Finally, a check is done to see if the inputs are both even, with an odd total amount of water needed. If this is met, then this will print out to the user that this is the reason the following operations won’t work.

The next loop is the main loop of the program, it fills the first bucket, dumps that into the second bucket until that bucket is bigger, because the larger bucket is always set to be jug b, we can safely make certain assumptions. After that dump, we empty that second bucket, then, in the next loop, check to see if we need to use a pour-to-fill or pour-to-empty algorithm. From there, either it starts running the first steps again if the first jug is now empty or empties the bucket and goes again.

The fill-until-empty and fill-until-full functions work simply, and check a few conditions first, once it validates the input, it takes the necessary water out of one bucket, and moves it to the other using a for loop. The fill and empty from and to the water source just remove all or fills to the total size of the jug.

The last two functions are crucial. trueCondition() checks if the combination of the two jugs is equal to the amount desired and prints a message, and quits if it is. This function, as well as the next, is called after every single use of one of the jug filling/emptying functions. The last major function prints the jug, but also does the checking to see if the function is repeating itself. It stores the current values in a string array, and every time it goes to print, it references these past values to see if the program is repeating itself. If it is, that means that the problem isn’t solvable, and the program exits.