Requirements:

1. The program must generate a random number between 1 and 10 (inclusive) (R, F)
2. The program must inform the user whether their guess is higher or lower than the generated number (R, F)
3. If they get the correct answer, the program must ask the user if they want to play again (R, F)
4. The program must generate only integers and accept only integers (R, NF)
5. The program must validate user input by making sure it's within the bounds given and continue after handling the error. (R, F)
6. The program must provide detailed instructions to the user (R, F)
7. The program can use pseudo random number generation (O, NF)
8. The program can be implemented in any programming language (NF)
9. The program must exit if the user provides an invalid input 3 times in one playthrough of the game (R, F)

Design:

1. Using C# an implemented as a console app because that is what I am the most comfortable with and will be able to implement most efficiently.
2. Begin by using a built-in random number generator to generate an integer in the range [1, 10]
3. Print terminal message explaining the functionality of the program and providing the instructions to the user.
4. Take input and validate that it is an integer in the range [1, 10]
    a. Print error message if the input is invalid and prompt them for another input including the rules for a valid input and increment of the doofus counter.
    b. If the input is accepted, then compare it to the originally generated number.
        i. If it is greater than the original number, then print "Lower"
        ii. If it is less than the original number, then print "Higher"
        iii. If they are equal print "Correct"
5. If the input is greater or less than then it should loop and prompt the user for another input.
6. If it is equal, then the loop should break and print a message asking if the user would like to go again.
7. If they say yes then loop from the beginning, if they say no then end the program.
8. A doofus counter will be set at the start of the first loop and reset on each loop of the game.

Requirements Traceability Matrix:

Requirements		Design Details								
	1	2	3	4	4.a	4.b	5	6	7	8	Total
1		F									F
2						F					F
3								P	P		F
4		P		P							F
5				P	P						F
6			P		P		P				F
7		F									F
8	F										F
9					P					P	F