Assignment 1 Design

Purpose:

The purpose of this program is to model a game called Left, Right, and Center. In this game, each player begins with 3 chips, and the goal is to be the last player remaining with chips. They take turns rolling a 6-sided die. Each turn, the player rolls the die as many times as the number of chips they have left. A 1, 2, or 3 means that the player keeps their chip. A 4 means that the player passes a chip to the person on the left. A 5 means that the player discards a chip to the pot in the middle. A 6 means that the player passes a chip to the person on the right. They keep on taking turns until only one player has chips. If a player does not have any chips left, they do not roll, but they are not eliminated because they could still receive chips from another player.

Questions:

Please answer the following questions before you start coding. They will help guide you through the assignment. To make the grader's life easier, please do not remove the questions, and simply put your answers below the text of each question.

Randomness:

Describe what makes randomness. Is it possible for anything to be truly random? Why are we using pseudorandom numbers in this assignment?

Randomness is created when a program is unpredictable. For this assignment, we are using srandom() and random() to make randomness. It is difficult to say if anything can be truly random. To our human eyes, yes, a computer can generate random numbers because we don't know where the starting point of the random number generator is. But since there has to be some fixed seed to start with, it technically isn't absolute randomness. In some sense it is possible for a quantum computer to generate true randomness, but this is a field I know very little about. We use a PRNG in this assignment because the results of our program must align with the autograder. This is a way to essentially generate controlled randomness, which can model a random result yet can be checked with the autograder.

What is an abstraction:

When writing code, programmers often use "abstractions". Define an abstraction in non computer science terms (Don't google it!)

An abstraction is a way to hide complicated aspects of something so that we only see a simplified representation of it. Sometimes when we have to see all the intricate details of something, it clogs are minds, and so it is easier to create a representation of them.

Why?:

The last assignment was focussed on debugging. How can abstractions make debugging easier? What other uses are there for abstractions? Hint: Do you have to be the one to write the abstraction?

As programmers, we often have lots of trains of thoughts that get mingled up with each other and we end up forgetting how we implemented things. Abstractions can make both you and other programmers understand what you were trying to do. Sometimes we may want to use some code from a library without being presented with all the details about them. An abstraction can make this process a whole lot easier.

Functions:

When you write this assignment, you can chose to write functions. While functions might make the program longer, they can also make the program simpler to understand and debug.

How can we write the code to use 2 functions along with the main? How can we use 8 functions? Contrast these two implementations along with using no functions. Which will be easier for you?

When you write the Program design section, think about your response to this section.

We can use two functions by having one generate a random number and another when somebody wins. We can use 8 functions by breaking apart the gameplay into its intricate details. Passing a token could be a function. Printing everything out could be a separate function. Or, we could use no functions outside of main, and everything can just run in a straight line. Honestly, for me personally, I feel like only using the main function would be the easiest, because this is not a complicated assignment and I don’t want to overcomplicate it.

Testing:

The last assignment was focused on testing. For this assignment, what sorts of things do you want to test? How can you make your tests comprehensive? Give a few examples of inputs that you will test.

I want to make sure that the output is the same as the reference program when everything is done correctly. I also want to check some other things that may not be as obvious. Such as, I want to make sure that 3 is the default number of players if an invalid response is given, and I want to make sure a default seed of 4823 gives the same result as the reference.

Putting it all together:

The questions above included things about randomness, abstractions and testing. How does using a pseudorandom number generator and abstractions make your code easier to test?

A PRNG will control the randomness to make sure that my code works properly, and I can compare it with the reference and receive proper results. Abstractions can make my code easier to debug, and so if a test identifies an error, I can easily see where I went wrong.

How to Use the Program:

Using this program is very simple. Just make sure you have lrc.c and the makefile on your computer. Compile your program using the make command, and you should have a binary file. Run ./lrc and it will prompt you to select the number of players. Choose a number between 3 and 10. If you don't it will use 3 by default. Then it will prompt you to input a seed. This is just a random number. If you don't input one, it will use 4823 by default. Then sit back and watch the game play out.

Program Design/functions

For this assignment, I plan to only use the main function. I will prompt the user to enter the number of players and a random seed, then store those numbers in variables. I will then get the players from the list of players and begin the game with a while loop. In that while loop, there will be a variable called player which changes after each turn. The number of rolls they get will be equal to the number of chips they have. They will then roll using a random number from 1-6. Each player will have a number of chips variable associated with them. A left roll will give one to the previous person, and a right roll will give one to the next person. This loop will continue through all the players, each time printing out how many chips they have left, until only one person has any chips left, in which case the loop ends and the winner is announced.