The towers of Hanoi are a classic recursion problem. The game involves 3 pegs and a number of disks of increasing size. The game starts with all disks on one peg in increasing order. The challenge is to move the stack to another peg obeying 3 rules. 1.) You can only move on disk at a time. 2.) You can only move a disk that is on the top of any given stack. 3.) A larger disk cannot be placed onto a smaller disk.

My “hanoi” function solves this problem for 3 pegs and ‘n’ disks. The “solveHanoi” function handles the user IO for the game. When running “solveHanoi” the user is given simple instructions and prompted to enter the name for each peg. The names are simply stored as Chars. It also prompts the user to input an integer value for the number of disks. The user IO function getDigit in Haskell will get a single char and return an Int. Once these have been passed in, “solveHanoi” will return “hanoi” with these parameters. The program will only take in the first number typed by the user. Any proceeding numbers will mess up the next call to “solveHanoi” if the user runs the program again. The output is a list of tuples that contain the correct moves in the correct order to solve the given tower.