University of Pennsylvania
EAS 105: Fall 2011
Matlab Project: Hanoi Puzzle (Part A)
Instructor: Donna Dietz

You should first read in your text/texts and/or on Wikipedia for relevant background material. Your classnotes should also prove useful. The play() method will not be tested using the tester, but it will be tested by hand, to see if the play is smooth. You may take liberties with this method to add your own personal touch! This project is pedagogically the first in a series of three projects. The second project will be a Matlab GUI for a Perpetual Calendar, and the third will be a GUI for this Hanoi Puzzle. The tester must run sufficiently quickly in order for your project to pass.

For this project create these files:

function [v] = int2baseTwo(x): x is a positive integer. Convert it to base two as a list of integers in reverse order. For example, int2baseTwo(6) = [0, 1, 1]. (Hint: dec2base)

function [v] = randLocationR(n): returns a random location (in rollcall format) within a size n Hanoi puzzle. This is returned as a vector of size (1,n) containing random digits between 1 and 3 inclusive. (Hint: find)

function [L] = rollcall2List(R): this returns a cellarray of 3 lists, representing the physical Hanoi puzzle. The integers 1 through n are distributed throughout the three sublists, and they are stored in increasing order within each sublist. For exapmle, $L = \{[1, 3, 5], [2, 4], [6]\}$ represents discs 1, 3, and 5 on the first post, discs 2 and 4 on the second post, and disc 6 (the largest) on the last post. Note that disc one is always on the top of some post.

function [boolean] = isLegalMove(L,a,b): returns True if it is legal to move a disc from post a to post b. L is the list of lists as described in the rollcall2List method. Returns False if move is illegal. If a==b, this may return either True or False.(This design decision will be yours to make.)

function [L] = makeMove(L,a,b): if it is legal to move the top disc from post a to post b, this method makes that move happen, otherwise it does nothing. The actual cellarray is actually changed and then returned. The cellarray must still be explictly changed in the calling code, however.

function printTowers(L): this prints the list of lists to the screen in a useful way for game play. The posts are labeled “1”, “2”, and “3”, and the display should be easy to understand. This method should be able to handle single and double digit values without making a mess of the display. (So, test it with a size 20 Hanoi game, for example.)

function [R] = list2Rollcall(L): this method takes in a cellarray of vectors as described earlier, and converts that into rollcall format. Rollcall always starts with the largest disc and ends with the smallest disc. For example, $L = \{[1, 3, 5], [2, 4], [6]\}$ as input would return $[3, 1, 2, 1, 2, 1]$

function [S] = rollcall2SA(R): this method takes in a Hanoi puzzle configuration in rollcall format and converts it to Sierpinski Address format. See class notes.

function [R] = SA2rollcall(SA): this method is the inverse function of rollcall2SA. Figure out how to accomplish this task! (Once you've gotten it, don't tell your classmates how you did it- that will ruin their fun!)

function [TA] = SA2TA(SA): this method takes in a Hanoi puzzle configuration as a Sierpinski Address and returns a Ternary Address for it. See class notes.

function [SA] = TA2SA(TA): this method is the inverse function of SA2TA. I leave this as a problem for the student. Hint: int2baseTwo

function [A,B] = reduceTA(A,B): this method takes in two Hanoi puzzle configurations in Ternary Address format and reduces it by "removing" any large dics which are in common. For example, if the 4 largest discs in both configurations are in the same locations, they are irrelevant to the problem. The two inputs are changed and returned by this method. Thus, the resulting two configurations will have their largest discs on two different posts. If the two inputs are equal, the vectors' contents are all set to zero.

function [dist] = distTA(A_in,B_in): If the two inputs are identical, 0 is immediately returned. Otherwise, the problem is reduced using the reduceTA method, and distance is calculated by the technique shown in the class notes.

function play(): allows the player to mess around with these fun puzzles! Rather than the usual game people typically play (which is to start with all the discs on one post and move them to another post), we will start with a random legal configuration, and try to get all discs on the final post. So, for small values of n, you may even get to start with a solved puzzle! The user should have a mechanism for quitting early, and of course, the fun part is getting the distance measure to work correctly! You should report the minimal distance to the solution after each move. Once your distance measure is working, it adds more fun and reduces stress in this game! You are encouraged to start writing this well before you have the distance measure finished, because you can use this method to help you debug earlier methods! If an illegal move is attempted, the player should be notified, but the configuration should not change. The user should be prompted for which post the disc should be taken from and which post the disc should be moved to. (As an optional variation, you may program the input pattern to be something else, such as "Which disc do you want to move?" and "Which disc should it be placed on top of?" However, this should be named something different such as play2().) Play should terminate when a puzzle is solved or when the user asks to quit. You may heckle the quitter as desired or congratulate the winner.