Artyom Martirosyan
10/21/2019
Assignment 4 design document

V.0

For this assignment we will be solving the Tower of Hanoi puzzle using 2 different methods: recursion and the stack. To start off I will create an int named num_rings which will hold the number of rings and set it to 5(default), I will then create a while loop and take in the command line arguments. From there I will have if statements one to send to the recursion function, another to send to the slack function. From there I will also divide num_rings by 2 and have an if statement if i get 1 the number is odd if 0 it is odd. In the recursion if statement I will create 2 arrays: a one dimensional one holding 3 chars for each poll, one 2d array [3] [numrings] which will hold the number of each ring (numbers will be allocated using a loop). These arrays will be developed inside of the if statement which will then refer to a function called recursion(sending the 2 arrays) the function will simply run a loop which will move all of the rings individually from keg A to keg C until only one ring is left in keg A. the program will then move the last ring onto keg B and call upon itself(this time changing the goal from moving all of the rings to keg B). I will also have another set of integers which will be used to check which keg is empty(this will help with moving the rings using the loop). The user will also create a counter which will count the number of moves taken which will print once keg B has num_rings. For the stack function I will be calling upon the stack and making 3 different stacks(one for each keg) and I will be running a loop to transfer the rings over to keg C(practically repeating the logic for the recursive function, but this time using the stack which will make it more simpler as you can check what the top ring or value is and work from there). With the stacks I am able to push and pop meaning that I can see the last thing that was pushed into the stack allowing for the code to compare the top values and work from there. (I personally feel like we should be able to rewrite the design pdfs after coding without losing points because in many cases(most of mine) I tend to learn as I play around with the code meaning that I can end up finding a more efficient method in order to execute the code, but it most likely will end up straying away from the initial design developed here.)

V.1

To start off the program will take in a user argument(if -n number provided change number of rings to said number), then the program will have two if statements if -r used run recursion method, if -s used run stack method. There will be several ints for both moves used and number of rings in each peg as well as arrays for the names of each peg and arrays for what's in each peg. From there we will start the two options: the recursion aspect of the program will start off with printing the title then it will run a loop which will fill up the arrays holding peg values with ring values, then it will have two if statements one for if odd or even(have two ints for odd and even and set them both to 5) if odd set ordering int to 0 if even do vice versa.Then it will run the function called recursion and send the 3 arrays through. Inside of the function there will be 3 if statements according to the position of ring one: the first position(zero) will send ring one from peg A to peg b(incrementing the ints used as stacks as well), there will then be an if statement where if B has reached maximum number of rings(user input - 1) then the function will

end. Then there will be an if statement for the other two pegs(without ring 1 on them) and it will decide which one has the smaller value(or nothing in it) and will send the smaller ring to the larger ring. You will have 2 more if statements with similar conditions(changing where peg one goes as well as which two other pegs will be compared) after these statements there will be 2 if statements(if odd move ring to right if even move left) and in the end the program will run itself again. This program will run until the if statement in the 1st ring position is met(also include this in the 3rd ring position as if even is chosen the ring will move to the left going from c to b hence filling the requirement). The other option is for using the stack where the same logic will be used for the most part except for the fact that the first loop which initializes values will only add them to stack a as keg a is the first one the rest will not be touched. After each move in both programs there will be a counter which will add 1 as the program goes on and will print out the final outcome. Also after each move the program will print saying which move is moved where using the array holding the names of the pegs. Note: the stack implementation headers are provided the user will need to implement the logic on their own and use stack rather than arrays for the stack option.