Aleesha Nageer
anageer@ucsc.edu
10/1/20

CSE 13S Fall 2020

Assignment 3: The Tower of Brahma

The Tower of Brahma (also known as Tower of Hanoi) moves disk between three rods, starting at one rod and ending at another. The disks are of different sizes and a larger disk cannot be placed on top of a smaller disk.

In this lab, I implemented two different methods of running this game: one recursive method and one stack-based method. I ensured all the rules were followed and used the getopt function to so that I can parse my arguments and the number of disks can vary each time the program is run.

The arguments for my program are:

1. -s: This will run the stack method
2. -r: This will run the recursive method
3. -n: This will take how many disks the game will be working with

TOP-LEVEL: *top level design of the code is given by the following pseudocode:*

```
Main:
    Flags for which argument is called
    Initialize Variable for Disk Number (n)

    Read program arguments
        Switch(method):
            Stack:
                Set Flag to True
            Recursion:
                Set Flag to True
            Number of Disks:
                If argument given:
                    Set n to Disk Number
                Else:
                    Set n to 5

    If Stack Flag True:
        Print Stack Header
        Run Hanoi Stack Function
    If Recursion Flag True:
        Print Recursion Header
        Run Hanoi Recursion Function
```

"Read Program Arguments" represents a getopt() loop that parses the arguments: s, r, and n

HANOI_RECURSION: *top level design of how I implemented the recursive method*

```
Hanoi_Recursion: (This is a void function that doesn't return anything. It prints with each call.)
    If Number of Disks = 1:
        Move disk from starting rod to ending rod

    Hanoi_Recursion (Decrement number of disks, move disk from starting rod to temporary rod)
    Increase move count
    Print ("Move disk # from starting rod to ending rod")
    Hanoi_Recursion (Decrement number of disks, move disk from temporary rod to ending rod)
```

HANOI_STACK: *top level design of how I implemented the stack-based method*

```
Hanoi_Stack: (This function returns an integer for the total number of moves made)
    Create Rods A, B, and C

    For (number of rods, decrement until all disks have been accounted for):
        Add disks to Rod A

    Initialize move and max_moves (2^n - 1) variables

    If Number of Disks is Odd:
        While current moves don't equal max_moves:
            If Rod B is empty or if Rod A isn't empty and the disk on A is greater
            than the disk on B:
                Move disk from A to B
            Else:
                Move disk from B to A

            Increment Moves
            Check if moves equals max_moves

            If Rod C is empty or if Rod A isn't empty and the disk on A is greater
            than the disk on C:
                Move disk from A to C
            Else:
                Move disk from C to A

            Increment Moves
            Check if moves equals max_moves

            If Rod C is empty or if Rod B isn't empty and the disk on B is greater
            than the disk on C:
                Move disk from B to C
            Else:
                Move disk from C to B

            Increment Moves
    Else (Even):
        While current moves don’t equal max_moves:
            If Rod C is empty or if Rod A isn’t empty and the disk on A is greater
            than the disk on C:
                Move disk from A to C
            Else:
                Move disk from C to A

            Increment Moves
            Check if moves equals max_moves

            If Rod B is empty or if Rod A isn’t empty and the disk on A is greater
            than the disk on B:
                Move disk from A to B
            Else:
                Move disk from B to A

            Increment Moves
            Check if moves equals max_moves

            If Rod B is empty or if Rod C isn’t empty and the disk on C is greater
            than the disk on B:
                Move disk from C to B
            Else:
                Move disk from B to C

            Increment Moves
```

To understand the pattern for even and odd I utilized the link, https://www.mathsisfun.com/games/towerofhanoi.html , to better understand how the game worked

I also came across a video online that helped me fully understand how I should go about creating my algorithm: https://www.youtube.com/watch?v=cZDydAdFrqg&t=327s

Design Process: *Modifications and problems I ran into*

I thought about the algorithm thoroughly for this lab, so I didn't end up changing my pseudocode for my design. However, I did run into problems where I had to spend time debugging and including print statements throughout my Hanoi_Stack function. This error was due to how I was using stack_peek and how I initially wrote the code for this helper function in my stack.c file. Once I noticed that I needed to decrement where my pointer was at by 1 to get the disk value, I was able to fix my code.

Overall, this lab was challenging but I learned so much about stacks and how to work with functions like pop, peek, and push. I got a deeper understanding of this ADT as well as reinforced my knowledge of recursive functions. Overall, creating this design and taking the time to understand how I would tackle this problem proved to be beneficial as I didn't need to change my pseudocode.