IE523 B,OB,ONL: Financial Computing
Fall, 2020
Programming Assignment 1: K Peg Version of the Tower of Hanoi Problem via Recursion
Due Date: 11 September, 2020

In Lesson 1 you were introduced to the *Frame-Stewart solution* for the 4 Peg version of the Tower Hanoi Problem. For this programming assignment, I want you to write a piece of C++ code that solves the *K Peg Version of the Tower of Hanoi Problem* that uses recursion.

That is, you have pegs $\{1, 2, \ldots, K\}$ (cf. figure 1). Peg 1 has n-many disks stacked in such a way that that any smaller diameter disk sits above all disks that are of larger diameter in peg 1. The objective is exactly the same as before – we want to find a way of transferring the stack of disks from peg 1 to peg K such that at no point in the process a larger disk sits on top of a smaller disk on a peg.

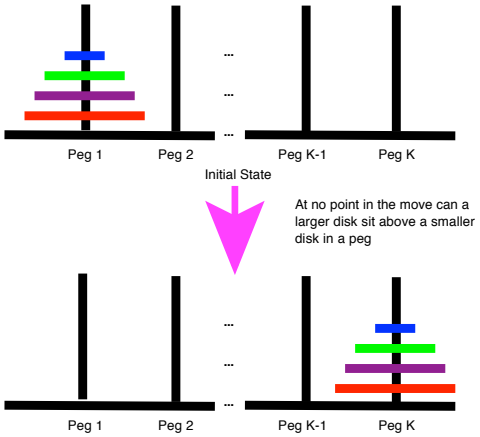

Figure 1: The *K Peg Version of the Tower of Hanoi* Problem.

This can be solved by a generalization of the Frame-Stewart Recursive Solution used in the previous programming assignment. That is, we want a recursive function $move(n, 1, K, \{2, 3, \ldots, K-1\})$, defined as

1. Let p be some number that is less than n. Move p disks from the top of peg 1 to the i-th peg, where $i \in \{2, 3, \ldots, K-1\}$ using $move(p, 1, i, \{2, 3, \ldots, K\} - \{i\})$.
2. Move the $n-p$ disks from peg 1 to peg K using $move(n-p, 1, K, \{2, 3, \ldots, K-1\} - \{i\})$. That is, this move does not involve/touch the i-th peg.
3. Move the p disks from peg i to peg K using $move(p, i, K, \{1, 2, \ldots, K-1\} - \{i\})$.

Here is what your program should do

1. Take as input, K, the number of pegs and the number of disks, n, of decreasing size, in peg 1.

2. Write a recursive routine that moves the n disks from peg 1 to peg K such that at no point during the moves, a larger disk sits on top of a smaller one in any peg using the generalized *Frame-Stewart Solution* described above. You have some leeway in deciding what p should be. I have found that
$$p = \begin{cases} \lfloor \frac{n}{2} \rfloor & \text{if } K - 2 > 1 \text{ (i.e. \#free pegs} > 1) \\ n - 1 & \text{otherwise.} \end{cases}$$
work quite well.

3. Your code should output (1) each individual move, and (2) the total number of moves in your solution.

4. Your implementation should have a function that checks if each move generated by your implementation is legal/valid. That is, for any value of K and n, you should check if no larger disk is placed on top a smaller disk.

You could start with generalizing the 3-peg solution that I wrote (cf. 3peghanoi.cpp and hanoi.h on Compass). I am looking for an output that looks like what is shown in figure 2. I have given you hints on Compass.

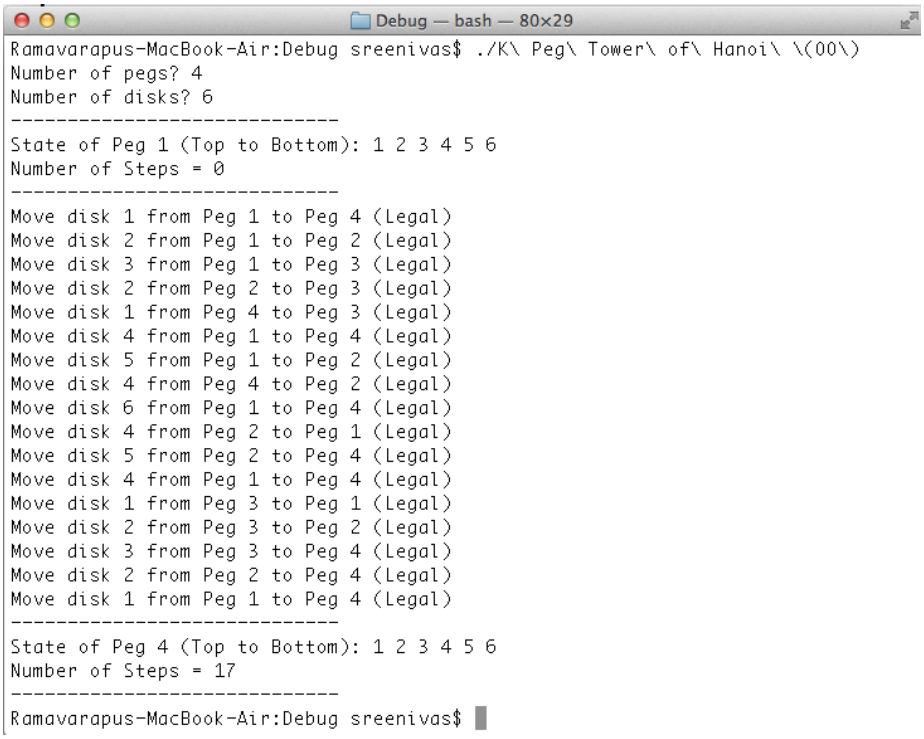

Figure 2: Sample Output.