In the following explanation, I will call the From_rod “rod A”, the help_rod “rod B”, and t
he to_rod “rod C”

Algorithm

Check if n meets the base case:
If n = 1, it will load the rods, showing which rods the disk will go from and to. Then, it will exit and go to the end function.
Go to if_end;

Recursively call the function:
Call function for moving n - 1 disks from rod A to rod C

Print out the moved disk from the rod A to the rod C

Call function for (n-1, rod B, rod C)

How I implemented the algorithm:

To solve the Tower of Hanoi problem, I used an algorithm that first moves n - 1 disks to the middle/help rod moves the biggest disk to the end rod, and then moves n -1 disks to the end rod. To do this, first I passed the parameters into the stack in the order with n-disks on top, then the rod these disks are moving from (rod A), and then the rod they are going to (rod C).
We first enter the parameters we want, in this case, we create the stack with (From top to bottom): n-disks = 3, rod A = 1, and rod C = 3, by pushing these elements into the stack.

The code then calls the function Hanoi which takes care of the recursive process to move the disks.
The code then saves the return address, makes the base pointer, and creates the rod B variables. We then load in the number of disks passed in and compare the value to 1. If it is equal, then the code loads in the values of the rod the disk will go from and the rod the disk will move to into registers 1 and 2, then the code jumps to the if_end function; if not, it executes the else function.

While the number of disks is not equal to 1, the code goes to the else function and first finds the position of the help rod. This is done by subtracting the amount for the to and from rod from 6 because the sum of all three is 6 (rod 1, rod 2, and rod 3). This will give you the value of the rod B. We then proceed to load the values of rod C and rod A, and we exchange the rod C spot with rod B by pushing in rod B as rod C and keeping rod A the same. After that, we will pass the amount of disks - 1 and call the function. We call function for (n-1, rod A, rod B) and then we usually move a disk from rod A to rod C and print it out. Then we do the same process and pass in rod B as rod A and rod C as rod C. We call the function for (n-1, rod B, rod C). This loop will continue until all the disks have been moved to rod C.

Role of the Stack Pointer:

The stack pointer is used to track the local variables used inside and outside the function and returns their addresses accordingly. In my code, I first save the link register to the stack, which is the address that the function will return. First, we pass in n-disks, rod A, and rod C. After each call to the functions, we pop the parameters to readjust the stack pointer.
The frame pointer is used to load the variables we want to use and push them into the function as the next parameter.
Each time there is a recursive call to the function, the function allocates memory for the rod B variable. When the function returns, the stack pointer restores to its original state.