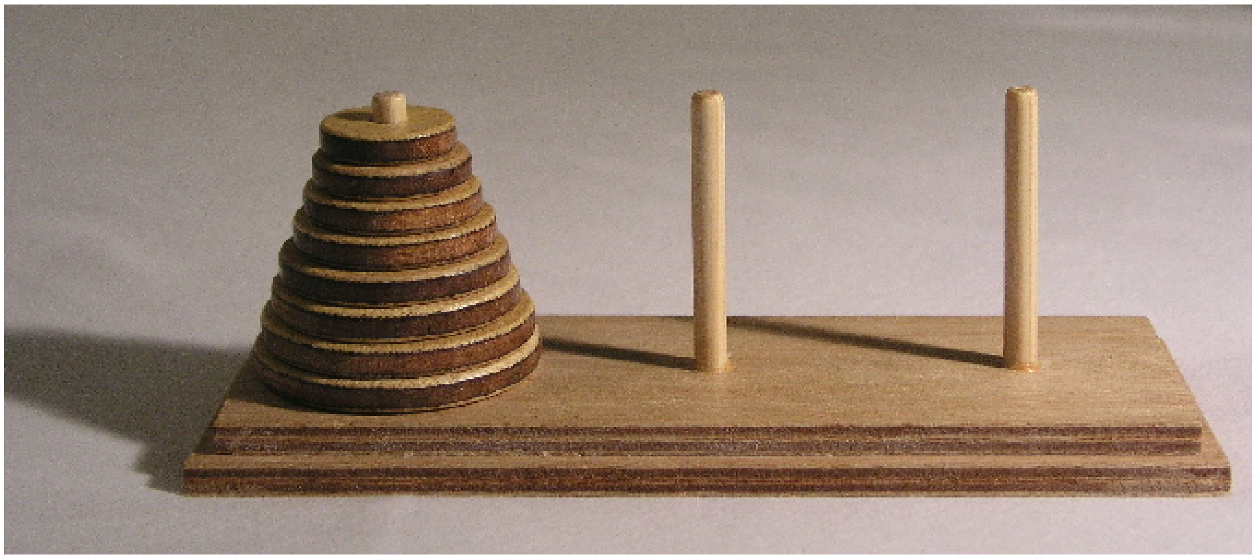
ISTA 350 Recursion Worksheet Name:

Write a function called `factorial` that recursively calculates the factorial of its nonnegative integer argument. The factorial of `0` or `1` is `1`, for `n > 1` it is `1 * 2 * … * n`. What is the big O of your solution? Create an iterative solution. What is its big O?

The rules of the Fibonacci sequence are: `F(0) = 0, F(1) = 1, F(n) = F(n - 1) + F(n - 2)`. Code up recursive and iterative functions that calculate `F(n)`. What are their big O's?


CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=228623

The Towers of Hanoi puzzle: move all of the disks from the starting pole, A, to the middle pole, B, using the righthand pole, C, as necessary. Rules: move one disk at a time from one pole to another, never putting a larger disk on a smaller disk. Write a recursive solution. This problem is hard to think about. Remember that recursion is great because it delegates the thinking. The only thing we have to figure is the base case and how to pass the buck of figuring it out to a recursive call. The function will take 4 arguments: the height of the stack of disks to move, the starting pole, the ending pole, and the helper pole. Delegate moving all of the disks but the bottom one to the helper pole. Move the bottom one to the ending pole (print the pole's starting and ending poles). Delegate moving the rest of the disks that are on the helper pole to the ending pole. What is the big O?

$$e^x = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots$$

Write a recursive function called `exp` that takes two arguments, float `x` and nonnegative integer `n`, and calculates an approximation to `e`^{x} by summing the first `n + 1` terms of the series. For example, to calculate the first six terms of `e`^{2}, add the terms shown above with the call `exp(2, 5)`.

Write function `third(precision, term)` that recursively calculates the partial sum 1/2 – 1/4 + 1/8 – 1/16 + 1/32 – … until `abs(term )` is less than `precision`. Sample calls: `third(0.001, 0.5),third(0.000008, 0.5)`.

Write a recursive function called `underscores` that takes a camelCase string and returns an under_score string. You may use a loop in your solution, but you must recurse every time you find an uppercase character. Example:

```
    underscore('iHeartRecursion')  -->  'i_heart_recursion'
```