Due: Wednesday Oct. 25th, 2017 at 10 am

Assignment Guidelines

- This assignment covers material in Module 05.
- Questions 1 and 2 must use only accumulative recursion. No correctness marks will be assigned otherwise. Questions 3 and 4 can use any form of recursion
- Submission details:
  - Solutions to these questions must be placed in files `a05q1.py`, `a05q2.py`, `a05q3.py`, and `a05q4.py`.
  - You must be using Python 3 or higher.
  - Download the interface file from the course Web page to ensure that all function names are spelled correctly and each function has the correct number and order of parameters.
  - All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
  - Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
  - For full style marks, your program must follow the Python section of the CS116 Style Guide.
  - Be sure to review the Academic Integrity policy on the Assignments page
- Download the testing module from the course web page. Include `import check` in each solution file.
- Restrictions:
  - Do not import any modules other than `math` and `check`.
  - Do not use any other Python functions not discussed in class or explicitly allowed elsewhere. See the allowable functions post on Piazza. You are always allowed to define your own helper functions, as long as they meet the assignment restrictions.
  - While you may use global *constants* in your solutions, do **not** use global *variables* for anything other than testing.
  - Read each question carefully for additional restrictions or tips.
  - Tip: for this assignment, repetition can be implemented using recursion as loops are not allowed

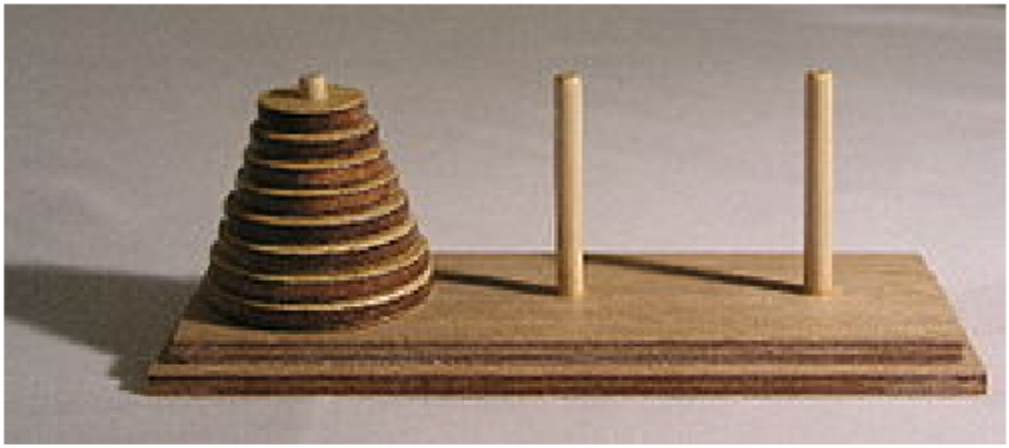
The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.

1. Note: You must use only accumulative recursion for this question.
   Write a Python function `calc_exp` that consumes three positive natural numbers `a`, `b` and `n`, and returns the result of
   $(b+b^2+b^3+....+b^n) + (a+1)(a+2).....(a+n)$
   For example,
   ```
   calc_exp(10,10,1) => 21
   calc_exp(10,3,4) => 24144 , because
   ```
   $3+3^2+3^3+3^4+(10+1)(10+2)(10+3)(10+4)=24144$

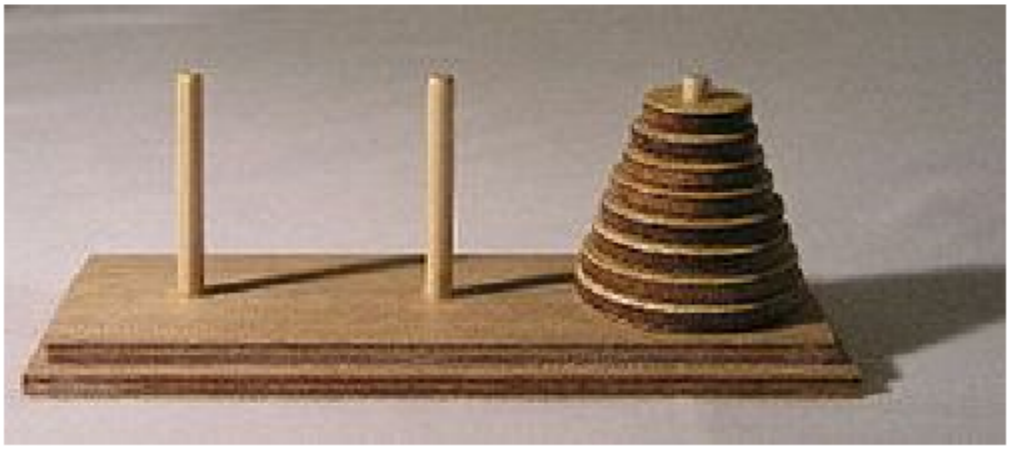
2. Note: You must use only accumulative recursion for this question.
   Write a Python function `list_stat` that consumes a list named `lst` and returns a list containing exactly 5 natural numbers in the order that follows:
   - The number of integers in `lst`
   - The number of floats in `lst`
   - The number of booleans in `lst`
   - The number of strings in `lst`
   - The number of all other types in `lst`

   For example,
   ```
   list_stat([3, "wow", -3.967, True, True, False, "nice"]) =>
   [1, 1, 3, 2, 0]
   list_stat(["good", [3,4], [10]]) => [0, 0, 0, 1, 2]
   ```

3. The Tower of Hanoi (also called the Tower of Brahma or Lucas' Tower, and sometimes pluralized) is a mathematical game or puzzle. It consists of three rods (source, helper, target), and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack (the source) in ascending order of size on one rod, the smallest at the top, thus making a conical shape. The objective of the puzzle is to move the entire stack to another rod (the target), obeying the following rules:
   - Only one disk must be moved at a time.
   - Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
   - No disk may be placed on top of a smaller disk.



A start point: A model set with 8 disks (three rods: source, helper, target)



The goal: moving the disks to the target rod following the above rules.

Write a function `show` that consumes, `n`, a natural positive number representing the number of disks, where the value of the numbers 1..`n` represent the size of the disks, `n` represents the disk in the bottom of the rod (the biggest disk) while 1 represents the disk on the top of the rod (the smallest disk). For example, `8` disks would be represented by the disks: `8,7,6,5,4,3,2,1`. The function prints the steps to solve the puzzle, in addition to the required number of steps to solve it. Note that the solution should be with minimum steps required to finish the game successfully.

For example, show(3) prints:
Moving 1 from source to target
Moving 2 from source to helper
Moving 1 from target to helper
Moving 3 from source to target
Moving 1 from helper to source
Moving 2 from helper to target
Moving 1 from source to target
The total number of steps required is: 7

(Note: there is one space between : and 7)

Advice: play the game manually several times with different number of disks before you start coding ☺
Here are links where you can play the game:
https://www.mathsisfun.com/games/towerofhanoi.html
http://www.dynamicdrive.com/dynamicindex12/towerhanoi.htm

4. Note: You are not allowed to use any of the string methods for this question (remember: ==, *slicing*, *in*, and *len* are not string methods, thus they are allowed).

Write a Python function `replace_str` that consumes 3 non-empty strings, `base`, `target` and `rep`. The first string, `base`, represents a base string that you want to update. The second string `target` represents the target string that you want to replace and the third string `rep` represents a string that will replace the target in the updated string. The function returns a new string in which the `target` string is replaced by the `rep` string in the `base` string, but returns the same `base` string if either of the following conditions hold true.

- If the `target` string is not found in the `base` string or,
- If the `target` and `rep` are the same strings.

For example,

```
replace_str("This is a book","a","the")=>'This is the book'
replace_str("This is my book","a","the")=>'This is my book'
replace_str("my brother reads books and sometimes he reads
magazines", "reads", "likes") => 'my brother likes books and
sometimes he likes magazines'
replace_str("Apple is a  fruit", "f" , "t")=>
'Apple is a truit'
replace_str("aaaaa","aa","x")=>'xxa'
```