Part A - Recursive functions

- Write the following python functions **recursively**.
- A non-recursive solution that works will not be given credit (even if it passes testing)

function 1:

This function is passed a number and returns

(number)(number-1)(number-2)...(3)(2)(1). By definition, 0! = 1

Only a recursive solution will be accepted!

```
def factorial(number)
```

function 2:

Write the **RECURIVE** function linear_search. linear_search() is passed a list of values and a key. If a matching key is found in the list, function returns index of where the key was found. If the key is not found, function returns -1.

NOTE: you are not allowed to use any of the built in list functions for this problem. The only function you are allowed to use is `len()`

```
def linear_search(list, key)
```

HINT: you may need to write the actual recursive function with a different set of arguments to accomplish this task.

function 3:

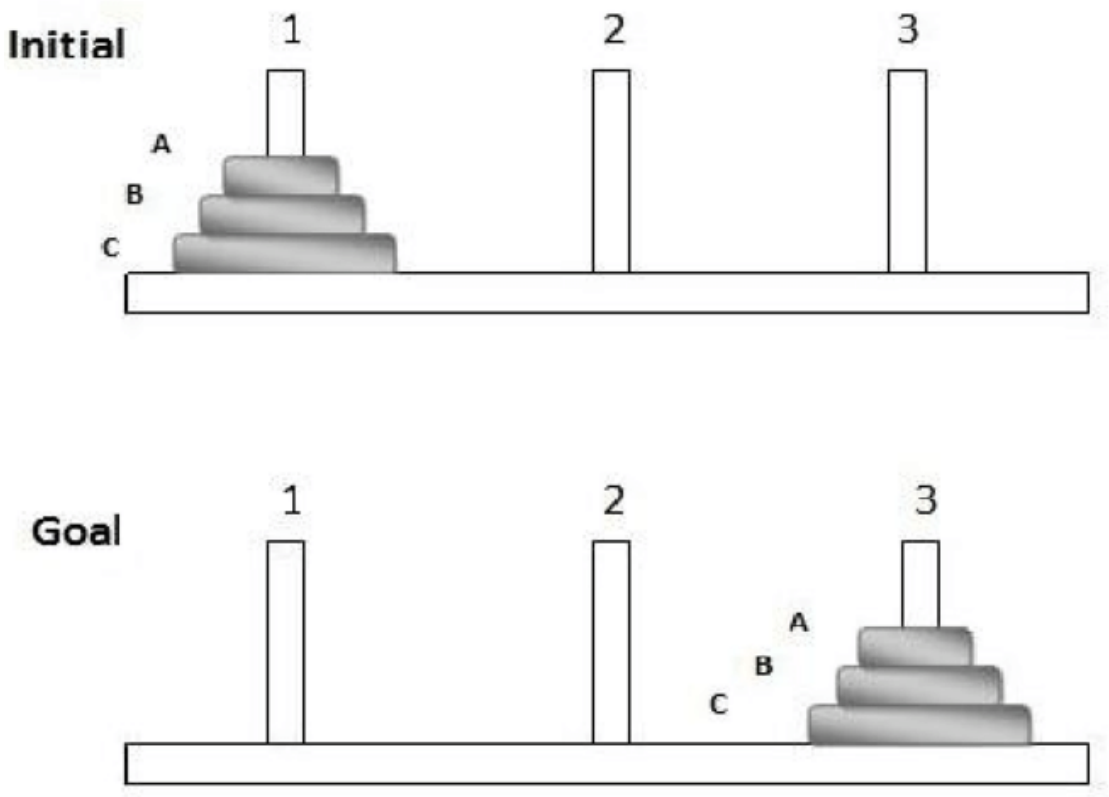
Write the **RECURIVE** function binary_search. `binary_search()` is passed a sorted list of values and a key. If a matching key is found in the list, function returns index of where the key was found. If the key is not found, function returns -1.

NOTE: you are not allowed to use any of the built in list functions for this problem. The only function you are allowed to use is `len()`

```
def binary_search(list, key)
```

HINT: you may need to write the actual recursive function with a different set of arguments to accomplish this task.

Function 4



The purpose is to move all the disks from tower 1 to tower 3 using tower 2. The rule is that no disk can be put on a smaller disk. In other words, the order of the disks should be maintained all through the transitions. Each move consists of movine ONE disk from a tower to the other. You can only move a disk if there is no other disk on top of it; otherwise the disk on top should be moved first. To summarize, here are the rules:

1. Only one disk can be moved at a time.
2. A disk can only be moved if it is the uppermost disk in the pole.
3. A larger disk can't be placed on a smaller disk.

Whrite a recusrive program that gets the number of disks on tower 1 and lists all the moves to take them to tower 3 using tower 2. The moves should be like:

1 to 2 (which means the top disk on tower 1 should be moved to tower 2)
1 to 3
2 to 3

Part B Analysis

Perform an analysis of the following recursive functions.

function 1:

Analyze the following function with respect to number

```
def function1(value, number):
	if (number == 0):
		return 1
	elif (number == 1):
		return value
	else:
		return value * function1(value, number-1)
```

function 2:

Analyze function2 with respect to the length of the mystring. Hint, you will need to set up two mathematical functions for operator counting. one for function2 and the other for recursive_function2

```
def recursive_function2(mystring,a, b):
	if(a >= b ):
		return True
	else:
		if(mystring[a] != mystring[b]):
			return False
		else:
			return recursive_function2(mystring,a+1,b-1)

def function2(mystring):
	return recursive_function2(mystring, 0,len(mystring)-1)
```

function 3 (optional challenge):

Analyze the following function with respect to number

```
def function3(value, number):
	if (number == 0):
		return 1
	elif (number == 1):
		return value
	else:
		half = number // 2
		result = function3(value, half)
		if (number % 2 == 0):
			return result * result
		else:
			return value * result * result
```