# W11.2 Applied

## Recursive Euclid’s Algorithm

Write a recursive function `rec_gcd(a, b)` that returns the greatest common divisor of two integers `a` and `b` using Euclid’s algorithm. Please see Week 2 Applied if you need a reminder of Euclid's Algorithm.

## List Inversion

Write a recursive function `reverse(lst)` that computes the reversion of `lst`. For example:

```
>>> reverse([1 , 2 , 3 , 4])
[4 , 3 , 2 , 1]
>>> reverse([10 , 11 , 12 , 13 , 14])
[14 , 13 , 12 , 11 , 10]
>>> reverse([1])
[1]
>>> reverse([])
[]
```

## Palindromes

Write a recursive function `is_pal(string)` to check if a `string` is palindrome. A palindrome string is a string that reads the same form either direction. For example:

```
>>> is_pal('aa')
True
>>> is_pal('aabb')
False
>>> is_pal('aba')
True
```

# Recursive Removal of Consecutive Duplicates

Discuss how to solve the following problem recursively.

```
def simplify(instring):
    """
    Input : a string (instring)
    Output : another string with the same content as instring but with all directly repeating chara
```

For example:

```
simplify('fffggsd') = 'fgsd'
simplify('abc') = 'abc'
simplify('') = ''
```

1. Discuss the problem and write down a recurrence relation that shows how an instance of the problem can be related to a simpler version of itself, i.e, an equation of the form:
   simplify(string) = ??
2. Can you find an alternative recurrence relation?
3. Also write down an equation for the base case. You are not meant to write code for this exercise but
   it may help in determining the recurrence relation.
4. Implement your solution in the Python file named `consecutive_duplicate_removal.py`.

## You are the Hanoi function!

In this week's workshop you have written a recursive function that solves the Towers of Hanoi puzzle. Form groups of 3 to 5 people to solve the puzzle collectively. If you are a group of $n$ people, start by setting up the puzzle in front of you in its initial state with $n$ disks on rod 0. You can

- use physical objects if available or
- draw it on the table with a whiteboard marker.

One person in the group is asked to move $n$ disks from rod 0 to rod 2. Every person now follows the following algorithm:

**When you are asked** to move $n$ disks from rod $i$ to rod $j$, let $k$ be the third (auxiliary) rod. **Do the following:**

- **If** $n = 1$
  - **move** a disk from rod $i$ to rod $j$
- **otherwise**
  - **ask** the person to your left to move $n - 1$ disks from rod $i$ to rod $k$
  - **move** a disk from rod $i$ to rod $j$
  - **ask** the person to your left to move $n - 1$ disks from rod $k$ to rod $j$

Each person should write their numbers $n$, $i$, $j$, and $k$ on the table in front of them to be sure to remember them. Erase them when you are done with your task and notify the person who asked you to do it.

Observe how the puzzle gets solved without any single person involved having an overview of the whole process.

## You are the function! (Recursively...)

Consider the following algorithm:

```
Algorithm StackSum(N)
... if N = 1...
... ... stackSoFar = create a stack with one element (eg. a lego block)
... ... return stackSoFar
... else...
... ... stackSoFar = StackSum(N-1)
... ... for i from 1 to N...
... ... ... add one block on top of stackSoFar
... ... return stackSoFar
```

Each person on your table will be a different version of StackSum. Your tutor will place some lego blocks or
equivalent on the table for your group (you can also do this task by drawing the stacks). One person on the
table will start and receive an $N$ value equal to the number of people on your table.

Each of you should:

- keep a sheet of paper where you track your own version of the variables $i$ and $N$
- when you are called on by another person, your $N$ value will become the value they pass you
- when you call another person, write down the value you are giving them, tear it off and hand it to them
- when you are returning, pass the lego stack back to the person who called you

# Feedback

**Question 1**

What worked best in this lesson?

*No response*

**Question 2**

What needs improvement most?

*No response*