

## Problem 1.

Similarly to what we had done in lecture, we can write the factorial function as follows:

```
def fact(n):
    if n == 0:
        return 1          # base case
    else:
        return n * fact(n-1) # recursive step
```

We constructed the *call tree* for `fact(3)` as follows:

```
fact(0)
  \
   /1
fact(1)
  \
   /1 * 1
fact(2)
  \
   /2 * 1 * 1
fact(3)
  \
   /3 * 2 * 1 * 1
```

Here, we write a function that computes  $x^n$  recursively, using only multiplication.

### Part (a)

How do you decompose  $x^n$  into smaller problems, in the same way we decomposed  $fact(n) = n \times fact(n-1)$ ?

What is the base case, if  $n$  is non-negative?

What is the recursive step?

### Part (b)

Write the function `power(x, n)` that computes  $x^n$  recursively, using only multiplication.

### Part (c)

Construct the call tree for `power(2, 3)`.

## Problem 2.

Write a function that sums up the digits of a number. For example, the sum of the digits of 123 is  $1+2+3 = 6$ . The function must be recursive.

Hint: what is the base case? What is the recursive step?

## Problem 2.

Without using loops, the `*` operator, or any function in the `math` module, write a function with the signature

```
times(a, b)
```

which returns  $a \times b$ .

Use recursion.

Use the fact that  $a \times b = (a \times (b - 1)) + a$ .

## Problem 3.

Using recursion, implement “linear search”: return the index of the element `e` in the list `L`. Do not use built-in functions or loops.

Hint: if `L[0]` is already `e`, you know the answer. If you know the index in `L[1:]`, you can transform it to be the answer for `L`.

## Problem 4.

Without using loops, write the function with the signature

```
interleave(L1, L2)
```

which takes `L1` and `L2`, two lists of the same length, and returns a list which consists of `L1` and `L2` interleaved, *i.e.*, `[L1[0], L2[0], L1[1], L2[1], ..., L1[n-1], L2[n-1]]` (here, `n == len(L1) == len(L2)`).

## Problem 5.

Without using loops or slicing, write a function that reverses a list in place. That is, the effect of calling

```
reverse_rec(L)
```

should be that `L` is reversed.

Here is how you might do this *with* loops:

```
def reverse_loop(L):
    for i in range(len(L)//2):
        L[i], L[-1-i] = L[-1-i], L[i]
```

You need to write a helper function that calls itself.

## Problem 6.

*Fun fact: this question is taken from the final exam that I wrote in my first year.*

Without using loops and without ever using `print` with a list (as opposed to individual elements of a list), write a function that, given a list `L` of size  $n$  (assume  $n$  is odd), prints the elements in the following order:

$L[n//2]$   $L[n//2-1]$   $L[n//2+1]$   $L[n//2-2]$   $L[n//2+2]$   $L[n//2-3]$   $L[n//2+3]$  ...  $L[n-1]$

Hint: write a recursive function that prints the following sequence for a list  $L$  of size  $n$ :

$L[0]$   $L[n-1]$   $L[1]$   $L[n-2]$   $L[n-3]$  ...  $L[n//2]$

```
def zigzag1(L):
    if len(L) == 0:
        print('')
    elif len(L) == 1:
        print(L[0], end = "")
    else:
        print(L[0], L[-1], end = "")
        zigzag1(L[1:-1])
```

## Problem 7.

This question is not meant to be about recursion.

In Project 3, you need to split up a string into sentences, with the sentences being separated by punctuation marks.

Here, you will do a similar problem with lists.

Write a function that splits up a list into sublists, where the sublist are elements of the original list separated by a given list of elements. For example, if the list is  $[1, 2, 6, 4, 5, 3, 7]$  and the list of elements is  $[3, 6]$ , the function should return  $[[1, 2], [4, 5], [7]]$ .

Hint: write a helper function that splits up a list into sublists separated by a single element. Then, use this helper function to split up the list by multiple elements. You can first replace e.g., every separator by the first element in the separator list, and then split up the list by that element.

Note: for strings, you can and should use the functions `str.split()` and `str.replace()`. However, for lists, you need to write your own function.

## Problem 8.

Work on Project 3.