UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

# CSE 15: Discrete Mathematics
# Laboratory 7

Fall 2020

- **This lab assignment will be graded and its grade will count towards the course grade**.
- **This lab must be solved individually.** You can discuss your ideas with others, but when you prepare your solution you must work individually. Your submission must be yours and yours only. No exceptions, and be reminded that the CSE academic honesty policy discussed in class will be enforced.
- Your solution must be exclusively submitted via CatCourses. Pay attention to the posted deadline because **the system automatically stops accepting submissions when the deadline passes. Late submissions will receive a 0**. You can upload one or more `.py` files.
- Start early.

## Introduction

In this lab you will write simple functions to implement some of the concepts related to recursion and induction that we have seen in class. To this end, it may be useful for you to refer and re-use any part of the code that was illustrated in the lab during the week November 2 to November 6 and available in Catcourses under the *Files* section.

## 1 Recursively defined function

Let $f : \mathbb{N} \to \mathbb{N}$ be a recursively defined function defined as follows:

$$f(n) = \begin{cases} 2 & \text{for } n = 0 \\ f(n-1) + 2 & \text{for } n \geq 1 \end{cases}$$

Write a Python function `compute_f` that given the parameter $n$ returns the value of $f(n)$. You function can either be recursive or iterative – the important thing is that it returns the correct result. You can assume that the parameter $n$ is indeed a natural number, i.e., inside the function you do not have to insert any code to check its validity.

## 2 Counting the number of occurrences of an element in a list

A list in python can have repeated elements. For example, the following statement creates a list including three 7s and two 9s.

```
L = [4,5,9,7,7,1,9,7,2]
```

Write a recursive function `count_instances` that given a list of integers `L` and an integer `n` returns the number of times that `n` occurs in `L`. For example, the function should return the following results:

```
count_instances(L,4)
>>>1
count_instances(L,7)
>>>3
count_instances(L,3)
>>>0
count_instances(L,9)
>>>2
```

To compute the results, the function should use the following recursive definition.

**Induction Basis** : if `L` is the empty list, then the result is always 0.

**Inductive Step** : if `L` is not the empty list, we distinguish two cases. If the first element of `L` is equal to `n`, then the result is 1 + `count_instances(tail(L),n)`, whereas if the first element of `L` is different from `n` then the result is `count_instances(tail(L),n)`. Here, `tail(L)` is the list obtained from `L` removing the first element. If `L` is not an empty list, in Python the tail of `L` can be obtained with the expression `L[1:]`.

To implement this function you have to check if `L` is empty or not. `L` is empty if it is equal to `[]` or if `len(L)` is 0 (the two conditions are equivalent and you can use either one.)