Laboratory Assignment 5

Problem 1 - nth frequency [15 pts]

Write a function that takes two arguments; a list and an integer called n. Return the list of numbers that repeats n times.

- Function parameter names do not matter, but the function should expect 2 parameters.
- inputs :
  - l = { x : x is list and len(x) ∈ [0:200]} and $l_i \in \mathbb{Z}$
  - n ∈ $\mathbb{Z}$
- output : list

```
>>> problem1([1, 2, 1, 1, 1, 2, 3, 4], 1)
[3, 4]

>>> problem1([1, 2, 1, 1, 1, 2, 3, 4], 0)
[]

>>> problem1([1, 2, 1, 1, 1, 2, 3, 4], -2)
[]

>>> problem1([1, 2, 1, 1, 1, 2, 3, 4], 2)
[2]

>>> problem1([1, 2, 1, 1, 1, 2, 3, 4], 3)
[]

>>> problem1([], 0)
[]

>>> problem1([], 1)
[]
```

Problem 2 - Median grade [15 pts]

Write a function that takes one argument: a dictionary. It will have the *name, grade* pair for the INF211 class. Find and return the median grade of the class.

- If the number of people is odd, return the middle grade.
- If the number of people is even, take the mean (average) of the two middlemost numbers and return that.
- if the dictionary is empty, return 0.
- Function parameter names do not matter.
- input : l = { x : x is dict and len(x) ∈ [0:1000]}
- output : integer or float

```
>>> problem2({'ahmet': 30, 'ali': 20, 'mehmet':10})
20
>>> problem2({'ahmet': 10, 'ali': 20})
15
```

Problem 3 - Black Box Testing [25 pts]

Write a function that takes two parameters: a function, and a number. This function's definition is given below, and the function may or may not work as expected. Your function should figure out if the given function is working or not working as expected. Return True if the function works correctly, False otherwise.

- Function parameter names do not matter.
- The function that will be tested takes one parameter, n, and should **find and return the number of 1's up to (including) the given number starting from 0.**
    - For example
        - ffunc(10) should return 2 since there are two 1's (1, and 10)
        - ffunc(100) should return 21 since there are 21 1's (1, 10, 11, 12, ..., 19, 21, 31, 41, ... 91, 100)
- `inputs :`
    - `ffunc = function that takes one parameter and returns the number of n's up to the given number (including) starting from 0.`
    - `n ∈ [0:300]`
- `output : Boolean`

```
>>> problem3(ffunc, 10)
True
>>> problem3(ffunc, 10)
False
```

Explanation: if ffunc implementation is correct (you need to test this to make sure it is) returns True, if ffunc implementation is incorrect, returns False.

Problem 4 - Combinations [25 pts]

Write a program that accepts a string and returns all possible combinations of the given strings.

- There should not be any duplicates in the returned list.
- The list should be ordered.
- Returned list should be lowercase.
- The words do not need to have a meaning.
- `inputs : s = { x : x is English letters and len(x) ∈ [1:10]}`
- `outputs: list of strings`

```
>>> problem4("a")
['a']
>>> problem4("AB")
['a', 'ab', 'b', 'ba']
>>> problem4("aba")
['a', 'aa', 'aab', 'ab', 'aba', 'b', 'ba', 'baa']
```

Problem 5 - Compression [20 pts]

Our engineers developed a novel compression algorithm that they claim will save a lot of space when saving text files. In this algorithm, if a letter is consequently repeated more than 1 time, instead of writing the letter multiple times, a number is appended to denote how many times the preceding letter is repeated. Write a program that compresses the given string, and returns the compressed version as a string and the **compression rate** as a percentage **round**ed to an integer.

- Function parameter names do not matter.
- **Compression rate** is described as the percentage of saved space.
- If the letter happens only 1 time, no number should be appended.
- `inputs : s = { x : x is English letters and len(x) ∈ [0:1000]}`
- `outputs: a tuple of (string, integer)`

```
>>> problem5('gol')
'gol', 0

>>> problem5('gollllllll')
'gol8', 60

>>> problem5('ggooooooooooollllllllllll')
'g2o12l13', 70

>>> problem5('ggooooooooooollllllllllllll')
'g2o12l14', 71
```

[1] https://raw.githubusercontent.com/dwyl/english-words/master/words_alpha.txt