CS-UY 1114 LAB 8, Spring 2018

In this lab, you will learn about functions, specifically:

- Functions - the syntax
- Passing in parameters
- Returning values
- Lists & indexing

Programming Questions:

Problem 1

Write a function that takes a lowercase character ch and a positive integer n as arguments. The function will return a string of n consecutive characters that begins with character ch. Wrap-around when applicable, i.e., consider 'a' be the next character to 'z'.

For example:

if ch = 'a' and n = 5, your function should return string 'abcde'.
if ch = 'x' and n = 6, your function should return string 'xyzabc'.

Problem 2

Write a function that takes two strings, a and b, as arguments. The function will return True if string a can be made from letters of string b. Note that each letter from string b can only be used once.

For example:

if a is 'aa' and b is 'abc', the function will return False because there's only 1 'a' in string b.
If a is 'ac' and b is 'aab', the function will return False because c is not in string b.
If a is 'abcdefg' and b is 'gfedcba', the function will return True because every letter that appeared in a also appeared in b.

Problem 3

Write a function that takes a string of 1s and 0s as argument. The function will prints out the number of consecutive 1s and 0s.

Example Output:

```
>>> encodeBinary('1111000110')
4 1's
3 0's
2 1's
1 0's
```

Problem 4

Write a **function** that takes an integer myInt and a positive integer n as two arguments and return a list of n consecutive integers that begins with myint.
For example:
If myInt = 10 and n = 5, your function should return the list [10, 11, 12, 13, 14].

Problem 5

Implement the **function** `count(lst, item)`. Returns the number of times item appears in lst.
For Example:
If lst is [0, 32, 'a', '0', '4', 15, 'q', '0'] and item is '0', it returns 2.

Problem 6

Create a list of the first n powers of 2. Ask user for n.
For Example:
If n = 4, create a list [2, 4, 8, 16]

Problem 7

The term *DNA sequencing* refers to methods for determining the order of the nucleotide bases, adenine, thymine, cytosine, and guanine in a molecule of DNA. The standard representation of the bases is to use their first letters, ATCG, so that DNA is represented as a string using only those four characters. However, DNA strings are millions of bases (characters) long.
Substring matching is the process of determining whether a shorter string (the substring) is contained within a longer string. Substring matching plays important roles in the reconstruction of an unknown DNA string from pieces and in searching for interesting substrings within a known DNA string.

Python provides a find(substring, start, end) string method that returns the lowest index (integer) where the substring is found in the index range start ≤ index < end. The start and end arguments are optional, but for this exercise we will make them required. If the substring is not found, -1 is returned.

(a) Without using the find string method, write a function that behaves exactly like the find string method. As your function is not a string method, the string to search must be an argument—let's make it the first one. The resulting format of your function will be: find(some_string, substring, start, end)

(b) Biology researchers frequently want to find *all* the locations where a substring is found, not simply the first one. Write a function named multi_find(some_string, substring, start, end) that, instead of returning one integer index, returns a list that contains zero or more indices separated by commas. In this case, the list will contain digits representing the integer indices. If the substring is not found, an empty list is returned.

HINT: Use the find function you created in part a!!!