Tutorial
Functional Programming

Question 1.

Let **lst** be a list of integer, write function **double(lst)** that returns the list of double of each element in **lst**
For example:
double([5,7,12,-4]) returns [10,14,24,-8]

a) Use list comprehension approach?

b) Use recursive approach?

c) Use high-order function approach?

Question 2.

Let **lst** be a list of a list of element, write function **flatten(lst)** that returs the list of all elements.
For example:
flatten([[1,2,3],['a','b','c'],[1.1,2.1,3.1]]) returns [1,2,3,'a','b','c',1.1,2.1,3.1]

a. Use list comprehension approach?

b. Use recursive approach?

c. Use high-order function approach?

Question 3.

Let **lst** be a list of integer and textbfn be an integer, write a function **lessThan**(n, lst) that returns a list of all numbers in **lst** less than **n**.
For example, lessThan(50, [1, 55, 6, 2]) returns [1,6,2]

a. Use list comprehension approach?

b. Use recursive approach?

c. Use high-order function approach?

Question 4.

Write function **compose** that can compose as many functions as you want. For example, there are three functions: **double**, **increase** and **square**. They can be called like **compose(double,increase) or compose(square,increase,double)**.

a. Use recursive approach?

b. Use high-order function approach?