National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2024/2025

Tutorial 6
Lists & Sequences

1. Ben Bitdiddle is required to implement a function `at_least_n` which takes in a list of integers and an integer n, and returns the **original** list with all the integers smaller than n removed.

```
>>> lst = list(range(10))
>>> lst2 = at_least_n(lst, 5)
>>> lst2
[5, 6, 7, 8, 9]
>>> lst is lst2
True
```

   (a) He implemented his function like this

```
def at_least_n(lst, n):
    for i in range(len(lst)):
        if lst[i] < n:
            lst.remove(lst[i])
    return lst
```

   Is this implementation correct?

   (b) Obviously the above implementation is wrong. But Ben decides to try again!

```
def at_least_n(lst, n):
    for i in lst:
        if i < n:
            lst.remove(i)
    return lst
```

   Is this implementation correct? What is the moral of the story?

   (c) Obviously the above implementation is also wrong. Help him out by implementing your own!

   (d) Now implement the function such that it returns a **new** list instead

```
>>> lst = list(range(10))
>>> lst2 = at_least_n(lst, 5)
>>> lst2
[5, 6, 7, 8, 9]
>>> lst is lst2
False
```

2. A matrix can be represented in Python by a list of lists (nested lists). For example, `m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]` represents the following 3×3 matrix:

$$\begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}$$

   (a) Write a function `col_sum` which takes in a matrix and returns a list, where the i-th element is the sum of the elements in the i-th column of the matrix, using nested `for` loops. You can assume that the matrix will not be empty, and has exactly $m \times n$ elements, where m and n are positive integers.

```
>>> m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> col_sum(m)
[12, 15, 18]
```

   (b) Write a function `row_sum` which takes in a matrix and returns a list, where the i-th element is the sum of the elements in the i-th row of the matrix.

   (c) Write a function `transpose` which takes in a matrix and transposes it. Basically, this converts an $m \times n$ matrix into an $n \times m$ matrix. The function should modify the **original** matrix.

3. Sort the list of integers `[5, 7, 4, 9, 8, 5, 6, 3]` into ascending order on paper. Show how the contents of the list evolves in each step. (This question is meant to ensure that you have a high-level understanding of the common sort algorithms. You are not required to write any code.)

   (a) Using **insertion sort**. (See mission):

   (b) Using **selection sort**. (See lecture):

   (c) Using **bubble sort**. (See recitation):

(d) Using **merge sort**. (See lecture or sidequest):

4. You are given a list of students in the following form (name, letter grades, score). For example,

```
students = [
    ('tiffany', 'A', 15),
    ('jane', 'B', 10),
    ('ben', 'C', 8),
    ('simon', 'A', 21),
    ('eugene', 'A', 21),
    ('john', 'A', 15),
    ('jimmy', 'F', 1),
    ('charles', 'C', 9),
    ('freddy', 'D', 4),
    ('dave', 'B', 12)]
```

The functions that you write for this question should work with any arbitrary list of students and not just for this sample list.

(a) Write a function `mode_score` that takes a list of students and returns a list of the mode scores (scores that appear the most number of times). If there is only one score with the highest frequency, then this list would only have one element.

For example:

```
>>> mode_score(students)
[15, 21]
```

(b) Write a function `top_k` that takes a list of students and an integer k and returns a list of the names of the k students with the highest scores in alphabetical order. If there are students in the range $(k+1, \cdots k+i)$ who have the same score as the kth student, include them in the list as well.

For example:

```
>>> top_k(students, 5)
[('eugene', 'A', 21), ('simon', 'A', 21), ('john', 'A', 15),
('tiffany', 'A', 15), ('dave', 'B', 12)]

>>> top_k(students, 3)
[('eugene', 'A', 21), ('simon', 'A', 21), ('john', 'A', 15),
('tiffany', 'A', 15)]
```