Assignment # 3: Arrays

1. Give one real-life example of how **arrays** can be used in real life. Then, describe or illustrate its concept and usage.

   For example, a book has several pages where we can record information on any page. You can write about a page has an order in which several pages form a sequence. You can also write about how each page can be represented by each cell in an array.

2. Given a 1-D array:

   Start Address is 1565,

   cell size is 9,

   What is the address of the element at index 10?

   Calculation steps should be provided.

3. Given a 1-D array:

   Start Address is 4120,

   address of the element at index 30 is 4450,

   What is the cell size of this array?

   Calculation steps should be provided.

4. Given a 2-D array of rank 3 of size 7:

   Start Address is 5050,

   element size is 7,

   What is the address of the element at index (3,6)?

   Calculation steps should be provided.

   Reminder: rank index starts at 0.

5. Tic-tac-toe is a game played on a 3-by-3 board. This can be represented by a 2-D array of rank 3, size 3. Two players 'X' and 'O' take turns in placing their marks in the cells on the board. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row or column is the winner.

   A 2-D array of tic-tac-toe call 'board' is as follows:

Index	**0**	**1**	**2**
0	(0,0)	(0,1)	(0,2)
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

   Write all winning conditions for this game by using 2-D array elements.

For example, board[0][0] == board[0][1] == board[0][2].

6. Given an array below:

9	10	5	7	3	1	0	2

6.1 Write the Python operations used and asymptotic running time *O*(?) executed for each operation to obtain the array below.

9	10	5	7	3	1		

6.2 From 6.1, write the Python operations used and asymptotic running time *O*(?) executed for each operation to obtain the array below.

9	10	7	3	1			

6.3 From 6.2, write the Python operations used and asymptotic running time *O*(?) executed for each operation to obtain the array below.

12	10	7	4	8	3	1	

6.4 From 6.3, write the Python operations used and asymptotic running time *O*(?) executed for each operation to obtain the array below.

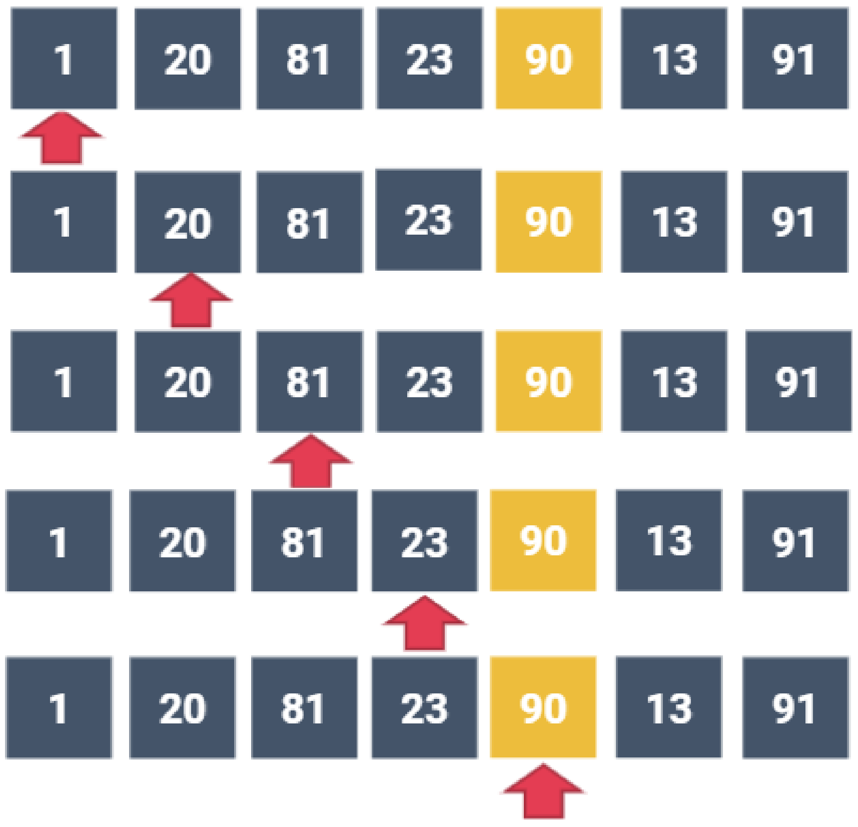
1	3	8	4	7	10	12	

6.5 From 6.4, write the Python operations used and asymptotic running time O(?) executed for each operation to obtain the array below.

1	3	4	7	8	10	12	

7. Write a Python code function of a linear (or sequential) search in a Python list.
    - function *linear_search* accepts two inputs: a list and an integer
    - It searches for an input integer in a list one-by-one starting from the first zero index.
        - If an element is found, returns the index of that element
        - Otherwise, it looks at the next index and repeatedly does this until the last element of the list.
    - If a list does not contain an input integer, return False

The example figure shows multiple steps to find "90" in an array and subsequently returns the corresponding index.



```
def linear_search(lst:list, x:int):

    ########################
    # Insert your code here
    ########################

    pass
```

8. Given an empty list *data* = [], record the <u>average</u> running time of insert(*k*, 20) in seconds with three different inserting patterns for each of the *N* calls:
    1. Repeatedly insert at the beginning of a list
    2. Repeatedly insert near the middle of a list
    3. Repeatedly insert at the end of the list

    Each pattern starts from an empty list

	N				
Patterns	100	1,000	10,000	100,000	1,000,000
First Case					
Second Case					
Third Case					