Laboratory Assignment 4

NOTE: Do NOT use `list` (or any other python specific keywords) as a parameter name, since this is a keyword, it will cause problems.

Problem 1 - Pick [4 pts]

Write a function that takes two arguments; a list with integer elements and an index, and return the indexed element from the list.

- Index starts from 0.
- If the index is `< 0` or `> length of the list` return `None`.
- Function parameter names do not matter, but the function should expect 2 parameters.
- `inputs :`
    - `l = { l : l is list and len(l) ∈ [0:200]} and` $l_i \in \mathbb{Z}$
    - `idx ∈` $\mathbb{Z}$
- `output :` l_{idx} `or NoneType`

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

>>> problem1([1, 2, 3], -1)
None

>>> problem1([1, 2, 3], 4)
None
```

Problem 2 - Remove [4 pts]

Write a function that takes two arguments; a list and an index, and return the list with the given indexed element removed from the list.

- Index starts from 0.
- If the index is `< 0` or `> length of the list` return the list as is.
- 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` $l_i \in \mathbb{Z}$
    - `idx ∈` $\mathbb{Z}$
- `output : list`

```
>>> problem2([], 2)
[]

>>> problem2([1, 2, 3], 2)
[1, 2]

>>> problem2([1, 2, 3], -1)
```

```
[1, 2, 3]

>>> problem2([1, 2, 3], 4)
[1, 2, 3]
```

Problem 3 - Sort [5 pts]

Write a function that takes two arguments; a list of numbers and a Boolean. If the Boolean value is True, return the list sorted in ascending order, if the Boolean value is False, return the list sorted in descending order.

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

```
>>> problem3([2, 1, 3], False)
[3, 2, 1]

>>> problem3([2, 1, 3], True)
[1, 2, 3]
```

Problem 4 - Common Numbers [8 pts]

Write a function that counts the numbers that are common in two given lists and returns that count.

- The numbers may repeat, but will still count as one.
- Function parameter names do not matter, but the function should expect 2 parameters.
- inputs : l1, l2 = { l : l is list and len(l) ∈ [0:200]} and $l1_i$, $l2_i \in \mathbb{Z}$
- output : integer

```
>>> problem4([1, 2, 3], [1, 3, 2])
3

>>> problem4([1, 2, 3], [2, 2, 2])
1

>>> problem4([1, 2, 3], [4, 5, 6])
0

>>> problem4([1, 2, 3], [3])
1

>>> problem4([1, 2, 3], [])
0
```

```
>>> problem4([1], [3, 5, 2])
0

>>> problem4([], [3, 5, 2])
0
```

Problem 5 - Weighted Average [8 pts]

Write a function that takes the weighted average of a given list. The function should expect two parameters as lists. The first list will be the numbers, and the second list will be the weights. The function should return the weighted average [1].

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

- Function parameter names do not matter, but the function should expect 2 parameters.
- inputs : l, w = { x : x is list and len(x) ∈ [1:200]} and $l_i, w_i \in R$ and len(l) == len(w)
- output : float

```
>>> problem5([1, 2, 3], [0.1, 0.2, 0.3])
2.3333333333333335

>>> problem5([1, 2, 3], [.5, .5, .5])
2.0
```

Problem 6 - Determinant [10 pts]

Write a function that will take a matrix in the form of a list of lists, find and return its determinant [2].

- Function parameter name does not matter, but the function should expect 1 parameter.
- If the size of rows and columns do not match (i.e. not a square matrix) return None.
- inputs : m = { x : x is list } and m_i = { x : x is list } and len(m) ∈ [1:4] and len(m_i) ∈ [1:4] and $m_{ij} \in \mathbb{Z}$
- output : float or NoneType

```
>>> problem6([[3, 7], [1, -4]])
-19.0

>>> problem6([[1]])
1.0

>>> problem6([[3, 7], [1, 2, 3]])
None
```

Problem 7 - Withdraw [18 pts]

Write a function that will withdraw money from a given account and return the updated accounts. The function should take four named arguments:

- `accounts:` a list that will hold the accounts
- `source:` the source index which the money will be withdrawn from
- `lira:` the amount that will hold the lira part of the withdraw
- `kurus:` the amount that will hold the kurus part of the withdraw

Accounts will be a list of strings that will hold all the accounts of the customers. The strings will be in the form of `lira.kurus` and you need to make appropriate conversions to do any operations on these strings. The final accounts list that is returned will also be in the same form. An example accounts list would be `["11.23", "10.43", "100.63", "0.10"]`.

- Index starts from 0.
- If the source is `< 0` or `> length of the list`, return `the accounts` with no change.
- If the transition is not doable (i.e. account has less money than the requested amount), return `the accounts` with no change.
- lira part of the string should NOT have any leading 0's. (i.e. should be "2" instead of "02")
- kurus part of the string should have the preceding 0's unless it is actually 0. If it is 0, just keep 0. (i.e. should be ".20" instead of ".2", ".40" instead of ".4", and ".0" instead of ".00")
- Function parameter names should be `accounts, source, lira, kurus`.
- `inputs :`
  - `accounts = { x : x is list of strings and len(x) ∈ [1:200]}`
  - `source ∈ ℤ`
  - `lira = { x : x ∈ [0:1E6]}`
  - `kurus = { x : x ∈ [0:99]}`
- `outputs : accounts = { x : x is list and len(x) ∈ [1:200]}`

```
>>> problem7(accounts=["11.23", "10.43", "100.63", "0.10"], source=-1, lira=10, kurus=13)
["11.23", "10.43", "100.63", "0.10"]

>>> problem7(["11.23", "10.43", "100.63", "0.10"], 1, 10, 13)
["11.23", "0.30", "100.63", "0.10"]

>>> problem7(["11.23", "10.43", "100.63", "0.10"], 4, 10, 13)
["11.23", "10.43", "100.63", "0.10"]

>>> problem7(["11.23", "10.43", "100.63", "0.10"], 3, 10, 13)
["11.23", "10.43", "100.63", "0.10"]
```

Problem 8 - Transfer [22 pts]

Write a function that will transfer money between given source and destination accounts and return the updated accounts. The function should take six named arguments:

- `accounts:` a list of strings that will hold the money on all the accounts

- `source:` the source index which the money will be transferring from
- `destination:` the destination index which the money will be transferring to
- `lira:` the amount that will hold the lira part of the transfer
- `kurus:` the amount that will hold the kurus part of the transfer
- `fee: True` or `False`. Should default to `False`. This will be used to apply transfer fees to the transaction or not.

Accounts will be a list of strings that will hold all the accounts of the customers. The strings will be in the form of "`lira.kurus`" and you need to make appropriate conversions to do any operations on these strings. The final accounts list that is returned will also be in the same form. An example accounts list would be `["11.23", "10.43", "100.63", "0.10"]`.

- Index starts from 0.
- The transfer amount should be subtracted from the source, and added to the destination.
    - If the `fee` parameter is set to `True`, the transfer fee should be subtracted from the source.
- The transfer fee should be `0.1` for any amount less than `10.0` liras, for any amount that is greater than `10.0` liras, the transfer fee should be `1%` of the transfer amount.
    - Careful with floating point operations here. (i.e. `1%` of `115` should be `1.15`, and not `1.1500000000000001`). Take the first two digits after the dot. (i.e. `0.1013` will be `0.10`)
- If the source and the destination indexes are the same, return `the accounts` list with no change.
- If the source or the destination are less than 0 or greater than the length of the list return `the accounts` list with no change.
- If the transition is not doable (i.e. account has less money than the requested amount) return the `accounts` list with no change.
    - If the `fee` parameter is set to `True`, and if the transition + the transfer fee is not doable (i.e. account has less money than the requested amount + the transfer fee) return `the accounts` with no change.
- `lira` part of the string should NOT have any leading 0's. (i.e. should be "2" instead of "02")
- `kurus` part of the string should have the preceding 0's unless it is actually 0. If it is 0, just keep 0. (i.e. should be ".20" instead of ".2", ".40" instead of ".4", and ".0" instead of ".00")
- Function parameter names should be `accounts, source, destination, lira, kurus, fee`.
- `inputs :`
    - `accounts = { x : x is list of strings and len(x) ∈ [1:200]}`
    - `source, destination ∈ ℤ`
    - `lira = { x : x ∈ [0:1E6]}`
    - `kurus = { x : x ∈ [0:99]}`
    - `fee ∈ {True, False}`
- `outputs : accounts = { x : x is list and len(x) ∈ [1:200]}`

```
>>> problem8(accounts=["11.23", "10.43", "100.63", "0.10"], source=0,
destination=1, lira=10, kurus=13)
["1.10", "20.56", "100.63", "0.10"]
```

```
>>> problem8(accounts=["11.23", "10.43", "100.63", "0.10"], source=0,
destination=0, lira=10, kurus=13)
["11.23", "10.43", "100.63", "0.10"]

>>> problem8(accounts=["11.23", "10.43", "100.63", "0.10"], source=3,
destination=0, lira=10, kurus=13)
["11.23", "10.43", "100.63", "0.10"]

>>> problem8(accounts=["11.23", "10.43", "100.63", "0.10"], source=0,
destination=1, lira=9, kurus=13, fee=True)
["2.0", "19.56", "100.63", "0.10"]

>>> problem8(accounts=["11.23", "10.43", "100.63", "0.10"], source=0,
destination=1, lira=10, kurus=13, fee=True)
["1.0", "20.56", "100.63", "0.10"]

Explanation: %1 of 10.13 is 0.1013. Taking the first two digits after dot
gives: 0.10.
```

Problem 9 - Bom 7 [13 pts]

In the class, you will be playing a game similar to Bom to determine the class representative. In this modified version, each student will have a number starting from 1, and you will count students in order, and wrap it around if it reaches the end. Every time you come to the 7th student, he/she will be eliminated, and the counting process will continue from the next student. Write a program that will accept the number of students in a class, and return the winner.

- Students' numbers start from 1.
- Function parameter name does not matter, but the function should expect 1 parameter.
- inputs : n = { x : x ∈ ℕ and 1E9 > x > 0}
- outputs: integer

```
>>> problem9(1)
1

>>> problem9(2)
2

>>> problem9(3)
3

Explanation: Assume three students: 1 2 3. We first start counting from 1.
```

- 1,2,3,1,2,3,**1**. The 1st student is the 7th, so he/she is eliminated. We then continue from where we left off. (2nd student)
- 2,3,2,3,2,3,**2**. The 2nd student is the 7th, so he/she is eliminated. Since there is only one student left (3rd), we return that number.

```
>>> problem9(4)
2

Explanation: Assume four students: 1 2 3 4. We first start counting from 1.
```

- 1,2,3,4,1,2,**3**. The 3rd student is the 7th, so he/she is eliminated. We then continue from where we left off. (4th student)
- 4,1,2,4,1,2,**4**. The 4th student is the 7th, so he/she is eliminated. We then continue from where we left off. (1st student)
- 1,2,1,2,1,2,**1**. The 1st student is the 7th, so he/she is eliminated. Since there is only one student left (2nd), we return that number.

Problem 10 - Mr. Lucky [8 pts]

Write a function that will find and return the only string that has a duplicate in a given list.

- Function parameter name does not matter, but function should expect 1 parameter.
- Return None if there is no duplicate.
- inputs : a = { x : x is list of strings and len(x) ∈ [0:200]}
- output : string or NoneType

```
>>> problem10(['1', '25', '36', '4', '1', '5'])
'1'

>>> problem10(['a','s','blasdf','dx','c','x','z','1','c','5'])
'c'

>>> problem10(['aba', 'hede', '1', 'aba'])
'aba'

>>> problem10([])
None

>>> problem10(['aba'])
None
```

[1] https://en.wikipedia.org/wiki/Weighted_arithmetic_mean
[2] https://en.wikipedia.org/wiki/Determinant