MICP Week 3 Homework

3sum

TEBOW IT

T phase: if no solution or [] or <3 in length, return []

E phase:

Sample Input	Equivalence class	Output
null	null	null
[]	empty	null/[]
[1,2,3,4,5,6]	No solution	null/[]
[-1,0,1]	One solution	[[-1,0,1]]
[0,1]	Not enough elements	null/[]
[0,0,0]	All zeros	[[0,0,0]]
[-1,0,1,2,-1,-4]	Multiple solutions	[[-1,0,1],[-1,-1,2]]

B phase:

- Go through every element in the array
- Check if -(arr[i] + arr[i+1]) is in the array

  eg: arr[i] = -1

  arr[i+1] = 0

  -(arr[i] + arr[i+1]) = 1

  so check if 1 is in the array
- If it is, add the three elements to the array to be returned since their sum is 0.
- While adding make sure the sub array is not already present in the array to be returned
- return the resultant array

O phase:

[-1, 0, 1, 2, -1, -4]

↑ ↑

-1 + 0 = -1

-(-1) = 1 which is in array

resulting array

[[-1, 0, 1],

[-1, -1, 2]]

0 + 1 = 1

-(1) = -1 which is in array but adding [0, 1, -1] would result in duplicate so don't add

1 + 2 = 3, -3 not in array

2 + (-1) = 1, -1 in array

-1 + (-4) = -5, 5 not in array

~ Think of time complexity

W phase:

If array is empty or length is less than 3 return []

For every element at i and i+1,

check if the negation of their sum is present in array

if it is, add all 3 elements to the array to be returned

else check other elements

while appending make sure that the sub array is not already present in the array to be returned.

if there is no solution, [] is returned.

I phase:

```
def threeSum(arr):
    if len(arr) <3 or arr == []:
            return []
    arrofArr =[]
    for i in range(0, len(arr)):

        if(-(arr[i] + arr[i+1])) in arr:
            subArr = [arr[i], arr[i+1],(-(arr[i] +arr[i+1]))]
            if (sorted(subArr)) not in arrofArr:
```

```
            arrOfArr.append(sorted(subArr))
    return arrOfArr
```

T phase: