```
# 1. https://leetcode.com/problems/richest-customer-wealth

class Solution(object):
    def maximumWealth(self, accounts):
        """
        :type accounts: List[List[int]]
        :rtype: int
        """
        wealth = []
        for account in accounts:
            total = 0
            for n in range(len(account)):
                total += account[n]
            wealth.append(total)

        high = wealth[0]
        for data in range(len(wealth)):
            if wealth[data] > high:
                high = wealth[data]

        return high



# 2. https://leetcode.com/problems/running-sum-of-1d-array/


class Solution(object):
    def runningSum(self, nums):
        """
        :type nums: List[int]
        :rtype: List[int]
        """
        total = 0
        for num in range(len(nums)):
            total += nums[num]
            nums[num] = total
        return nums




# 3. https://leetcode.com/problems/jewels-and-stones


class Solution(object):
    def numJewelsInStones(self, jewels, stones):
        """
        :type jewels: str
        :type stones: str
        :rtype: int
        """
        out = 0
        for n in stones:
            if n in jewels:
                out += 1

        return out
```

```
# 4. https://leetcode.com/problems/minimum-absolute-difference

class Solution(object):
    def minimumAbsDifference(self, arr):
        """
        :type arr: List[int]
        :rtype: List[List[int]]
        """
        full = []
        arr = sorted(arr)
        # print(arr)
        value = arr[-1]-arr[0]
        for pos in range(len(arr)-1):
            diff = arr[pos+1]-arr[pos]
            # print(arr[pos+1],arr[pos], diff)
            if diff <= value:
                if diff < value:
                    full = []
                value = diff
                data = [arr[pos], arr[pos+1]]
                full.append(data)
                # print(full)
        return full

# 5. https://leetcode.com/problems/three-consecutive-odds

class Solution(object):
    def threeConsecutiveOdds(self, arr):
        """
        :type arr: List[int]
        :rtype: bool
        """
        odd = []
        out = False
        for n in arr:
            if n%2 != 0 and n != 2:
                odd.append(n)
                if len(odd) == 3:
                    out = True
                    break
            else:
                odd = []
        return out

# 6. https://leetcode.com/problems/transpose-matrix

class Solution(object):
    def transpose(self, matrix):
        """
        :type matrix: List[List[int]]
        :rtype: List[List[int]]
        """
        full = []
        for n in matrix:
            for i in range(len(n)):
                if len(full)-1 < i:
                    full.append([n[i]])
                else:
```

```
                data = full[i]
                data.append(n[i])
        return full



# 7. https://leetcode.com/problems/majority-element

class Solution(object):
    def majorityElement(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
        value = len(nums) / 2.0
        datalist = []
        for n in nums:
            if n not in datalist:
                datalist.append(n)
                counted = nums.count(n)
                if counted > value:
                    return n


# 8. https://leetcode.com/problems/move-zeroes

class Solution(object):
    def moveZeroes(self, nums):
        """
        :type nums: List[int]
        :rtype: None Do not return anything, modify nums in-place instead.
        """
        non_zero = 0
        for n in range(len(nums)):
            if nums[n] != 0:
                nums[non_zero] = nums[n]
                non_zero += 1

        for i in range(non_zero, len(nums)):
            nums[i] = 0

        return nums
```