Question-1

Reverse a part of array (arr, start, end) Then solve https://leetcode.com/problems/rotate-array/description/ (https://leetcode.com/problems/rotate-array/description/)

```C++
class Solution {
public:
    void rotate(vector<int>& nums, int k) {
        // Brute force O(n*k) SC:O(1)

        1 2 3 4

        4 1 2 3
        3 4 1 2
        2 3 4 1
        1 2 3 4

        2%4 = 2
        4%4 = 0

        // Using additional array TC:2*O(N) = O(N)  SC: O(N)

        // 1,2,3,4,5,6,7 k = 3, n=7
        // reverse first n-k elements, reverse last k elements
        // 4 3 2 1 7 6 5
        // 5 6 7 1 2 3 4
        // TC: O(N) SC: O(1)

        reverse() // first n-k elements

        reverse() // last k elements

        reverse() // for entire array

        // reverse first k elements, reverse last n-k elements -> left rotation
        // 3 2 1 7 6 5 4
        // 4 5 6 7 1 2 3
    }

    void reverse(vector<int>& nums, int s, int e) {

    }
};
```

In [7]:
```python
from typing import List
class Solution:
    def reverse(self, start, end, arr):
        while(start<end):
            arr[start],arr[end] = arr[end],arr[start]
            start+=1
            end-=1
        return arr

    def rotate(self, nums: List[int], k: int) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        l = len(nums)
        k %= l
        self.reverse(0,l-k-1,nums)
        self.reverse(l-k,l-1,nums)
        self.reverse(0,l-1,nums)
        return(nums)

s = Solution()
a = [1,2,3,4,5,6]
s.rotate(a, 2)
print(a)
```

```
[5, 6, 1, 2, 3, 4]
```

```
JavaScript
/**
 * @param {number[]} nums
 * @param {number} k
 * @return {void} Do not return anything, modify nums in-place instead.
 */
var rotate = function(nums, k) {
       const n = nums.length;
        k=k%n;

        reverse(nums,0,n-k-1);
        reverse(nums,n-k,n-1);
        reverse(nums,0,n-1);
};

function reverse(nums, s, e)
    {
        while(s<e)
        {
            const temp=nums[s];
            nums[s]=nums[e];
            nums[e]=temp;
            s++;
            e--;
        }
    }
```

In [ ]:

Question-2

https://leetcode.com/problems/trapping-rain-water/ (https://leetcode.com/problems/trapping-rain-water/)

In [ ]:

In [ ]:
```
// 2,1,0,1,3,2,1,2,1
// Brute force: TC: O(N^2)  SC: O(1)
// for each element: // O(N)
        l = find max on left // O(N)
        r = find max on right // O(N)
        currentWaterLevel = min(l,r)
        if currentWaterLevel > currBarHeight:
            water += currentWaterLevel - currBarHeight


// TC: O(N)  SC: O(N)
// maxonLeft = array
// compute maxOnLeft // O(N)
// maxonRight = array
// compute maxonRight // O(N)

// for each element: // O(N)
        currentWaterLevel = min(maxOnLeft[i],maxonRight[i])
        if currentWaterLevel > currBarHeight:
            water += currentWaterLevel - currBarHeight



class Solution:
    def trap(self, height: List[int]) -> int:
        lrAuxArray = [0 for _ in range(len(height))] # max on left
        rlAuxArray = [0 for _ in range(len(height))] # max on right

        tempMax = height[0]
        for index in range(len(height)):
            tempMax = max(tempMax, height[index])
            lrAuxArray[index] = tempMax

        tempMax = height[len(height) - 1]
        for index in range(len(height) - 1, -1, -1):
            tempMax = max(tempMax, height[index])
            rlAuxArray[index] = tempMax

        water = 0
        for index in range(len(height)):
            water += min(lrAuxArray[index], rlAuxArray[index]) - height[index]

        return water

// 2,1,0,1,3,2,1,2,1
// 0 1 2 3 4 5 6 7 8
// 2 2 2 2 3 3 3 3 3 maxOnLeft
// 3 3 3 3 3 2 2 2 1 maxOnRight
// 0 1 2 1 0 0 1 0 0 => 5
```

In [ ]:
```
class Solution:
    def trap(self, height: List[int]) -> int:
        rlAuxArray = [] # max On right

        tempMax = height[len(height) - 1]
        for index in range(len(height) - 1, -1, -1):
            tempMax = max(tempMax, height[index])
            rlAuxArray = [tempMax] + rlAuxArray

        tempMax = 0
        leftMax = tempMax # calculate maxOnLeft on the fly
        for index in range(len(height)):
            leftMax = max(leftMax, height[index])
            tempMax += min(leftMax, rlAuxArray[index]) - height[index]

        return tempMax
```

In [8]:

```
l = 0
r = n - 1
maxOnLeft = height[0]
maxOnRight = height[r]
total = 0
while(l < r) {
    if (height[l] < height[r]) {

        if (height[l] >maxOnLeft ) {
            maxOnLeft = height[l]
        } else {
            total += (maxOnLeft - height[l])
        }
        l++
    } else {

        if (height[r] >maxOnRight ) {
            maxOnRight = height[r]
        } else {
            total += (maxOnRight - height[r])
        }
        r--
    }
}
Arr 1 2 1 3 2 4 1
    0 1 2 3 4 5 6

l = 0   1 2 3 4 5
r = 6 5
L = 3
R = 1

total = 0 + 0 + 1 + 1

```

```
  File "C:\Users\LEANGA~1\AppData\Local\Temp/ipykernel_5544/2157692379.py", line 6
    while(l < r) {
                 ^
SyntaxError: invalid syntax
```

In [ ]:

Question-3
https://leetcode.com/problems/longest-palindrome/

In [ ]:
```
TC: O(N)
SC: O(N)

class Solution {
public:
    int longestPalindrome(string s) {
        "abbbbbaccc"
        a : 2
        b : 6
        c : 4

        // count frequency
        freqmap <char, int>
        for (i = 0; i < s.size(); i++) {
            if c in freqmap:
                freqmap[c] += 1
            else
                freqmap[c] = 1
        }

        totalLen = 0
        oddChar = 0
        for key, value in freqmap:
            if value % 2 == 0:
                totalLen += value
            else:
                totalLenn += value - 1
                OddChar = 1

        return totalLen + hasOddChar

    }
};
```

In [ ]:

```Python
class Solution:
    def longestPalindrome(self, s: str) -> int:
        freq = {}
        res = 0
        for i in s:
            if i in freq:
                freq[i] += 1
            else:
                freq[i] = 1

        odd_char = ''
        for k, v in freq.items():
            if v%2 == 1:
                v = v-1
                odd_char = k

            res += (k*v)
        print('res = ', res, 'odd_char = ', odd_char)

        return res + odd_char + res[::-1]

class Solution:
    def longestPalindrome(self, s: str) -> int:
        d = {}
        for i in s:
            if i in d:
                d[i] += 1
            else:
                d[i] = 1

        odd_included = False
        l = 0
        for i in d:
            val = d[i]//2
            l += val*2
            if(odd_included==False and d[i]%2):
                odd_included = True
                l += 1

        return(l)


```

In [14]:
```
def longestPalindrome(s: str) -> int:
    freq = {}
    for i in s:
        if i in freq:
            freq[i] += 1
        else:
            freq[i] = 1

    res = ''
    odd_char = ''
    for k, v in freq.items():
        if v%2 == 1:
            v = v-1
            odd_char = k

        res += (k*(v//2))
    print('res = ', res, 'odd_char = ', odd_char)

    return res + odd_char + res[::-1]

longestPalindrome('abbbcdaabd')
```

```
res =  abbd odd_char =  c
```

Out[14]:
```
'abbdcdbba'
```

In [13]:
```

```

Out[13]:
```
'a'
```

In [ ]:
```

```

In [ ]:
```
n = len(nums)
k = k % len(nums)

l,r = 0,n-k-1
while l<r:
    nums[l],nums[r] = nums[r],nums[l]
    l,r = l + 1,r-1

l,r = (n-k),len(nums)-1
while l<r:
    nums[l],nums[r] = nums[r],nums[l]
    l,r = l+1,r-1

l,r = 0,len(nums)-1
while l<r:
    nums[l],nums[r] = nums[r],nums[l]
    l,r = l + 1,r-1
return nums
```