ARRAYS PART 3

(1) ROTATE ARRAY

```
def rotate_array(nums, k):
    n = len(nums)
    k = k % n

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

def reverse(nums, start, end):
    while start < end:
        nums[start], nums[end] = nums[end], nums[start]
        start += 1
        end -= 1

nums = [1, 2, 3, 4, 5, 6, 7]
k = 2
rotate_array(nums, k)
print(nums)
```

(2)BEST TIME TO BUY AND SELL STOCK 2

```
class Solution:
    def maxProfit(self, prices: List[int]) -> int:
        maximumprofit = 0
        for i in range(1, len(prices)):
            if prices[i] > prices[i - 1]:
                maximumprofit += prices[i] - prices[i - 1]
        return maximumprofit

prices = [7,1,5,3,6,4]
print(stock2(prices))
```

(3) HAPPY NUMBER LEETCODE

```
class Solution:
  def isHappy(self, n:int) -> bool:
    seen = set()
    while n != 1 and n not in seen:
      seen.add(n)
      n = self.sum_of_squares(n)
    return n == 1

  def sum_of_squares(self, num:int) -> int:
    total = 0
    while num > 0:
      total += (num % 10) ** 2
```

```
      num //= 10
    return total

n = 19
print(Solution().isHappy(n))
```

(4)NEXT PERMUTATION

```
class Solution:
    def nextPermutation(self, nums: List[int]) -> None:
        """
        Do not return anything, modify nums in-place instead.
        """
        n = len(nums)
        i = n - 2
        while i >= 0 and nums[i] >= nums[i + 1]:
            i -= 1
        if i >= 0:
            j = n - 1
            while nums[j] <= nums[i]:
                j -= 1
            nums[i], nums[j] = nums[j], nums[i]
        left = i + 1
        right = n - 1
        while left < right:
            nums[left], nums[right] = nums[right], nums[left]
            left += 1
            right -= 1

  return nums
nums = [1, 2, 3, 6, 5, 4]
print(nexpermutation(nums))
```

(5)REMOVE DUPLICATES FROM SORTED ARRAY 2

```
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:
        if len(nums) <= 2:
            return len(nums)
        i = 2
        for j in range(2,len(nums)):
            if nums[j] != nums[i - 2]:
                nums[i] = nums[j]
                i += 1
        return i
```

(6) LONGEST PALINDROMIC SUBSTRING

```
def longest_palindrome(s):
  res = ""
  reslen = 0
  #even length
  for i in range(len(s)):
    l, r = i, i
    while l >= 0 and r < len(s) and s[l] == s[r]:
      if (r - l + 1) > reslen:
        res = s[l:r + 1]
        reslen = r - l + 1
      l -= 1
      r += 1

    #odd length
    l, r = i, i + 1
    while l >= 0 and r < len(s) and s[l] == s[r]:
      if(r - l + 1) > reslen:
        res = s[l:r + 1]
        reslen = r - l + 1
      l -= 1
      r += 1
  return res

s = "babad"
print(longest_palindrome(s))
```

(7)