```
def removeele(nums,val):
    k=0
    for i in range(len(nums)):
        if nums[i]!=val:
            nums[k]=nums[i]
            k+=1
    return k
nums=[2,3,2,3,4]
val=3
k=removeele(nums,val)
print(nums[:k])



def maxsubarraysum(a,size):
    max_so_far=a[0]
    current_max=a[0]
    for i in range(1,size):
        current_max=max(a[i],current_max+a[i])
        max_so_far=max(max_so_far,current_max)
    return max_so_far
a=[-2,1,-3,4,-1,2,1,-5,4]
print(maxsubarraysum(a,len(a)))

import itertools
p=itertools.permutations([1,1,2])
unique=list(dict.fromkeys(list(p)))
print([list(perm)for perm in unique])

def len_of_last_word(s:str)->int:
    s=s.rstrip()
    words=s.split()
```

```
    return len(words[-1])if words else 0
input_str="Hello world"
print(len_of_last_word(input_str))


def isValidSudoku(board):
    rows = [set() for _ in range(9)]
    columns = [set() for _ in range(9)]
    boxes = [set() for _ in range(9)]

    for r in range(9):
        for c in range(9):
            if board[r][c] == '.':
                continue
            if board[r][c] in rows[r] or board[r][c] in columns[c] or board[r][c] in boxes[(r // 3) * 3 + (c //
3)]:
                return False
            rows[r].add(board[r][c])
            columns[c].add(board[r][c])
            boxes[(r // 3) * 3 + (c // 3)].add(board[r][c])

    return True
board = [
    ["5","3",".",".","7",".",".",".",".","."],
    ["6",".",".","1","9","5",".",".",".","."],
    [".","9","8",".",".",".",".",".","6","."],
    ["8",".",".",".",".","6",".",".",".","3"],
    ["4",".",".","8",".","3",".",".",".","1"],
    ["7",".",".",".",".","2",".",".",".","6"],
    [".","6",".",".",".",".",".","2","8","."],
    [".",".",".",".","4","1","9",".",".","5"],
    [".",".",".",".",".","8",".",".","7","9"]
```

```
]
print(isValidSudoku(board))

def combinationSum(candidates, target):
    dp = [[] for _ in range(target + 1)]
    dp[0] = [[]]
    for c in candidates:
        for i in range(c, target + 1):
            dp[i] += [comb + [c] for comb in dp[i - c]]
    return dp[target]
candidates = [2, 3, 6, 7]
target = 7
print(combinationSum(candidates,target))

def getPermutation(n, k):
    from math import factorial
    nums = list(range(1, n + 1))
    k -= 1
    result = []
    for i in range(n, 0, -1):
        idx, k = divmod(k, factorial(i - 1))
        result.append(nums.pop(idx))
    return ''.join(map(str, result))
n = 3
k = 3
print(getPermutation(n, k))

def countAndSay(n):
    if n == 1:
        return "1"
```

```
    def nextSequence(s):
        result = []
        i = 0
        while i < len(s):
            count = 1
            while i + 1 < len(s) and s[i] == s[i + 1]:
                i += 1
                count += 1
            result.append(str(count) + s[i])
            i += 1
        return "".join(result)

    current = "1"
    for _ in range(n - 1):
        current = nextSequence(current)
    return current
print(countAndSay(4))
```