main.py

Save Run

Output

```
REMOVE ELEMENT

n=[1,2,3,4,4]
k=4
while k in n:
    n.remove(k)
print(n)
```

```
[1, 2, 3]

=== Code Execution Successful ===
```

```
MAXIMUM SUBARRAY
def c(a):
    csum=a[0]
    msum=a[0]
    for n in a[1:]:
        csum=max(n,n+csum)
        msum=max(msum,csum)
    return msum
a=[-2,1,-3,4,-1,2,1,-5,4]
print(c(a))
```

```
6

=== Code Execution Successful ===
```

COUNTANDSAY

```python
class Solution:
    def countAndSay(self, n: int) -> str:
        if n==1:
            return "1"
        prev_say = self.countAndSay(n-1)
        say = ""
        count = 1
        for i in range(len(prev_say)):
            if i == len(prev_say)-1 or prev_say[i] != prev_say[i+1]:
                say += str(count) + prev_say[i]
                count = 1
            else:
                count += 1
        return say
n = 1
print(Solution().countAndSay(n))
```

```
1

=== Code Execution Successful ===
```

COMBINATION SUM

```python
class Solution:
    def combinationSum(self, candidates, target):
        def backtrack(remain, comb, start):
            if remain==0:
                result.append(list(comb))
                return
            elif remain<0:
                return
            for i in range(start, len(candidates)):
                if i > start and candidates[i] == candidates[i - 1]:
                    continue
                comb.append(candidates[i])
                backtrack(remain - candidates[i], comb, i)
                comb.pop()
        result = []
        backtrack(target, [], 0)
        return result
candidates = [2, 3, 6, 7]
target = 7
print(Solution().combinationSum(candidates, target))
```

```
[[2, 2, 3], [7]]

=== Code Execution Successful ===
```