1.Count and Say.

main.py

```python
def countAndSay(n):
    if n==1:
        return "1"
    prev=countAndSay(n-1)
    result= ""
    i=0
    while i<len(prev):
        count=1
        while i+1<len(prev) and prev[i]==prev[i+1]:
            i+=1
            count+=1
        result+=str(count)+prev[i]
        i+=1
    return result
print(countAndSay(5))
```

Output

```
111221

=== Code Execution Successful ==
```

Time complexity: O(nlogn)

2.Maximum Subarray.

main.py

```python
def maxSubArray(nums):
    if not nums:
        return 0
    max_sum=float('-inf')
    current_sum=0
    for num in nums:
        current_sum=max(num,current_sum+num)
        max_su=max(max_sum,current_sum)
    return max_sum
nums=[-2,1,-3,4,-1,2,1,-5,4]
print(maxSubArray(nums))
```

Output

```
6

=== Code Execution Successful
```

Time complexity: O(n^2)

3.Remove element.

main.py Save Run Output

```python
def removeElement(nums,val):
    if not nums:
        return 0
    i=0
    for j in range(len(nums)):
        if nums[j]!=val:
            nums[i]=nums[j]
            i+=1
    return i
nums=[3,2,2,3]
val=3
result=removeElement(nums,val)
print("Input array: ",nums)
print("Output array: ",result)
```

```
Input array:  [2, 2, 2, 3]
Output array:  2

=== Code Execution Successf
```

Time complexity: O(n)

4.Permutations.

main.py Save Run Output

```python
def permuteUnique(nums):
    def backtrack(start, end):
        if start == end:
            result.append(nums[:])
        for i in range(start, end):
            if i > start and nums[i] == nums[start]:
                continue
            nums[start], nums[i] = nums[i], nums[start]
            backtrack(start + 1, end)
            nums[start], nums[i] = nums[i], nums[start]
    result = []
    nums.sort()
    backtrack(0, len(nums))
    return result
nums = [1, 1, 2]
print(permuteUnique(nums))
```

```
[[1, 1, 2], [1, 2, 1], [2

=== Code Execution Succes
```

Time complexity: O(n^2)