```
import math
def getPermutation(n,k):
    nums=[str(i) for i in range(1,n+1)]
    k-=1
    res=""
    while n>0:
        n-=1
        index=k//math.factorial(n)
        k%=math.factorial(n)
        res+=nums.pop(index)
    return res
n=3
k=3
output=getPermutation(n,k)
print(output)



def max_subarray(nums):
    max_sum=current_sum=nums[0]
    for num in nums[1:]:
        current_sum=max(num,current_sum+num)
        max_sum=max(max_sum,current_sum)
    return max_sum
nums=[-2,1,-3,4,-1,2,1,-5,4]
print(max_subarray(nums))



def combinationSum(candidate,target):
    result=[]
    def backtrack(remaining,combination,start):
        if remaining==0:
```

```
            result.append(list(combination))
            return
        elif remaining<0:
            return
        for i in range(start,len(candidates)):
            combination.append(candidate[i])
            backtrack(remaining-candidates[i],combination,i)
            combination.pop()
    backtrack(target,[],0)
    return result
candidates=[2,3,6,7]
target=7
print(combinationSum(candidates,target))

def removeelement(nums,val):
    writepointer=0
    for readpointer in range(len(nums)):
      if nums[readpointer]!=val:
          nums[writepointer]=nums[readpointer]
          writepointer+=1
    return writepointer
nums1=[2,4,7,1]
val1=7
k1=removeelement(nums1,val1)
print(k1,nums1[:k1])



def combinationSum(candidates,target):
    def backtrack(start,target,path):
        if target==0:
            result.append(path)
```

```
            return
        if target<0:
            return
        for i in range(start,len(candidates)):
            if i>start and candidates[i]==candidates[i-1]:
                continue
            backtrack(i+1,target-candidates[i],path+[candidates[i]])
    candidates.sort()
    result=[]
    backtrack(0,target,[])
    return result
candidates=[10,1,2,7,6,1,5]
target=8
print(combinationSum(candidates,target))

def permuteUnique(nums):
    def backtrack(start):
        if start == len(nums):
            result.append(nums[:])
            return
        lookup = set()
        for i in range(start, len(nums)):
            if nums[i] in lookup:
                continue
            lookup.add(nums[i])
            nums[start], nums[i] = nums[i], nums[start]
            backtrack(start + 1)
            nums[start], nums[i] = nums[i], nums[start]

    nums.sort()
    result = []
```

```
    backtrack(0)
    return result
nums = [1, 1, 2]
print(permuteUnique(nums))

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