1021. Remove Outermost Parentheses

```
class Solution:
    def removeOuterParentheses(self, S: str) -> str:
        counter = 0
        result = []
        for i in S:
            if i == '(':
                counter += 1
                if counter > 1:
                    result.append(i)
            else:
                counter -= 1
                if counter > 0:
                    result.append(i)
        return ''.join(result)
```

523. Continuous Subarray Sum

We will get reminder of presum % k. TC is O(n), SC is O(k)

```
class Solution:
    def checkSubarraySum(self, nums: List[int], k: int) -> bool:
        mem = {}
        pre_sum = 0
        mem[0] = -1
        for idx, num in enumerate(nums):
            pre_sum += num
            if k != 0:
                pre_sum = pre_sum % k

            if pre_sum in mem:
                if idx - mem[pre_sum] > 1:
                    return True
            else:
                mem[pre_sum] = idx
        return False
```

560. Subarray Sum Equals K

We will use presum, pretty easy.

```
from collections import defaultdict
class Solution:
    def subarraySum(self, nums: List[int], k: int) -> int:
        res, mem, pre_sum = 0, defaultdict(int), 0
        mem[0] = 1
```

```
        for num in nums:
            pre_sum += num
            res += mem[pre_sum - k]
            mem[pre_sum] += 1
        return res
```

325. Maximum Size Subarray Sum Equals k
We will use presum and hashmap to solve this question. TC is O(n), SC is O(n)

```
class Solution:
    def maxSubArrayLen(self, nums: List[int], k: int) -> int:
        longest_dis, mem, pre_sum = 0, {}, 0
        mem[0] = -1
        for idx, num in enumerate(nums):
            pre_sum += num
            if pre_sum - k in mem:
                longest_dis = max(idx - mem[pre_sum - k], longest_dis)
            if pre_sum not in mem:
                mem[pre_sum] = idx
        return longest_dis
```

53. Maximum Subarray
The key here is to maintain the minimum pre_sum and sum(subarray) = max(cur_pre_sum - min_pre_sum), TC is O(n), SC is O(1)

```
class Solution:
    def maxSubArray(self, nums: List[int]) -> int:
        cur_min, cur_max = 0, -float('inf')
        pre_sum = 0
        for num in nums:
            pre_sum += num
            cur_max = max(pre_sum - cur_min, cur_max)
            cur_min = min(pre_sum, cur_min)
        return cur_max
```