16. 3SUM CLOSEST

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.

Return *the sum of the three integers*.

You may assume that each input would have exactly one solution.

CODE:

```
class Solution(object):
    def threeSumClosest(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: int
        """
        nums.sort()
        closest_sum = float('inf')
for i in range(len(nums) - 2):
    left, right = i + 1, len(nums) - 1
    while left < right:
        current_sum = nums[i] + nums[left] + nums[right]
        if abs(current_sum - target) < abs(closest_sum - target):
            closest_sum = current_sum
```

```
            if current_sum < target:
                left += 1
            elif current_sum > target:
                right -= 1
            else:
                return current_sum

    return closest_sum
```

TESTCASE:

nums =[-1,2,1,-4]

target =1

17.LETTER COMBINATIONS OF A PHONE NUMBER

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

CODE:

```
class Solution:
    # @return a list of strings, [s1, s2]
    def letterCombinations(self, digits):
        if '' == digits: return []
        kvmaps = {
            '2': 'abc',
            '3': 'def',
            '4': 'ghi',
            '5': 'jkl',
            '6': 'mno',
            '7': 'pqrs',
            '8': 'tuv',
            '9': 'wxyz'
        }
        return reduce(lambda acc, digit: [x + y for x in acc for y in kvmaps[digit]], digits, [''])
```

TESTCASE:

digits=”23”

18. 4Sum

Given an array nums of n integers, return *an array of all the* ***unique*** *quadruplets* [nums[a], nums[b], nums[c], nums[d]] such that:

- 0<= a, b, c, d <n
- a, b, c, and d are **distinct**.
- nums[a] + nums[b] + nums[c] + nums[d] == target

You may return the answer in **any order**.

CODE:

```
class Solution(object):
    def fourSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[List[int]]
        """
        ans = []
        n = len(nums)
        if n < 4:
            return ans

        nums.sort()
        for i in range(0, n - 3):
            if i != 0 and nums[i] == nums[i - 1]:
```

```
            continue
        for j in range(i + 1, n - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            k = j + 1
            g = n - 1
            while k < g:
                total = nums[i] + nums[j] + nums[k] + nums[g]
                if total < target:
                    k = k + 1
                elif total > target:
                    g = g - 1
                else:
                    temp = [nums[i], nums[j], nums[k], nums[g]]
                    ans.append(temp)
                    k = k + 1
                    g = g - 1
                    while k < g and nums[k] == nums[k - 1]:
                        k = k + 1
                    while k < g and nums[g] == nums[g + 1]:
                        g = g - 1
        return ans
```

TESTCASE:

nums=[1,0,-1,0,-2,2]

target =0

19.Remove nth node from end of list

Given the head of a linked list, remove the n^{th} node from the end of the list and return its head.

CODE:

```
class Solution:
    def removeNthFromEnd(self, head, n):
        def index(node):
            if not node:
                return 0
            i = index(node.next) + 1
            if i > n:
                node.next.val = node.val
            return i
        index(head)
        return head.next
```

TESTCASE:

head=[1,2,3,4,5]

n =2

20.Valid parentheses

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

CODE:

```
class Solution(object):
    def isValid(self, s):
        """
        :type s: str
        :rtype: bool
        """
        stackdict={
            ')':'(',
            ']':'[',
            '}':'{'
        }
        stack=[]
        for i in s:
            if i in stackdict.values():
```

```
            stack.append(i)
        if i in stackdict.keys():
            if stack and stack.pop()==stackdict[i]:
                pass
            else: return False
    if stack: return False
    else: return True
```

TESTCASE:

S=”()”