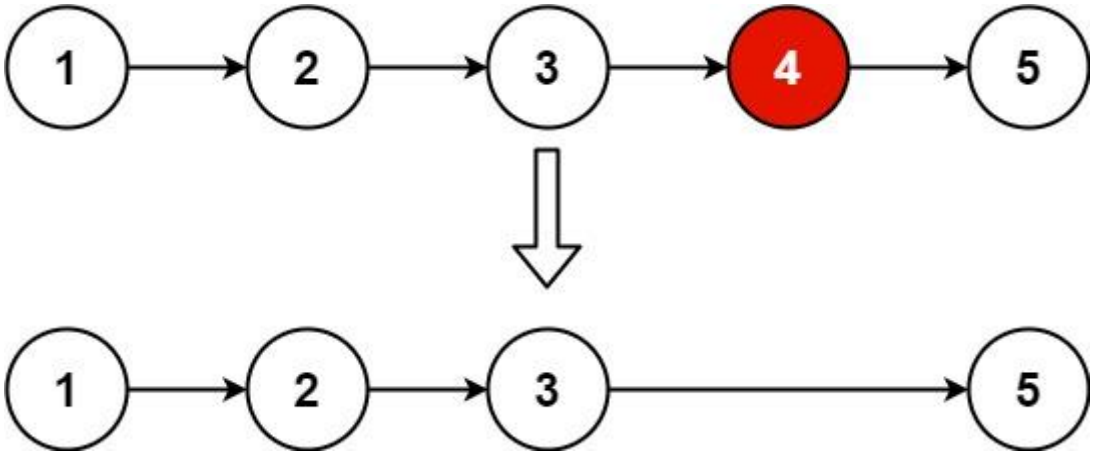
[19. Remove Nth Node From End of List(1)]
Given the head of a linked list, remove the n_{th} node from the end of the list and return its head.

Example 1:



Input: head = [1,2,3,4,5], n = 2
Output: [1,2,3,5]

Example 2:

Input: head = [1], n = 1
Output: []

Example 3:

Input: head = [1,2], n = 1
Output: [1]

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def removeNthFromEnd(self, head: Optional[ListNode], n: int)
-> Optional[ListNode]:
        dummy = ListNode(0)
        dummy.next = head
        slow = fast = dummy
        for _ in range(n+1):
            fast = fast.next
        while fast:
            fast = fast.next
            slow = slow.next
        slow.next = slow.next.next
        return dummy.next
```

[69. Sqrt(x)(2)]

Given a non-negative integer x, return *the square root of* x *rounded down to the nearest integer*. The returned integer should be **non-negative** as well.

You **must not use** any built-in exponent function or operator.

- For example, do not use pow(x, 0.5) in c++ or x ** 0.5 in python.

Example 1:

Input: x = 4
Output: 2
Explanation: The square root of 4 is 2, so we return 2.

Example 2:

Input: x = 8
Output: 2

Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.

```
class Solution:
    def mySqrt(self, x: int) -> int:
        if x<2:
            return x
        l = 1
        r = x//2
        while l<=r:
            mid = (l+r)//2
            if mid * mid == x:
                return mid
            elif mid*mid <x:
                l = mid+1
            else:
                r = mid -1
        return r
```

[367. Valid Perfect Square(3)]

Given a positive integer num, return true *if* num *is a perfect square or* false *otherwise*.

A **perfect square** is an integer that is the square of an integer. In other words, it is the product of some integer with itself.

You must not use any built-in library function, such as sqrt.

Example 1:

Input: num = 16
Output: true
Explanation: We return true because 4 * 4 = 16 and 4 is an integer.

Example 2:

Input: num = 14

Output: false
Explanation: We return false because 3.742 * 3.742 = 14 and 3.742 is not an integer.

```
class Solution:
    def isPerfectSquare(self, num: int) -> bool:
        if num == 1:
            return True
        l = 1
        r = num//2
        while l<=r:
            mid = (l+r)//2
            if mid*mid == num:
                return True
            elif mid*mid <num :
                l = mid+1
            else:
                r = mid - 1
        return False
```

633. Sum of Square Numbers(4)

Given a non-negative integer c, decide whether there are two integers a and b such that $a^2 + b^2 = c$.

Example 1:

Input: c = 5
Output: true
Explanation: 1 * 1 + 2 * 2 = 5

Example 2:

Input: c = 3
Output: false

```python
class Solution:
    def judgeSquareSum(self, c: int) -> bool:
        l = 0
        r = int(c**0.5)
        while l<=r:
            current_sum = l *l + r * r
            if current_sum == c :
                return True
            elif current_sum < c:
                l = l+1
            else:
                r = r - 1
        return False
```

50. Pow(x, n) (5)

Implement pow(x, n), which calculates x raised to the power n (i.e., x^n).

Example 1

Input: x = 2.00000, n = 10
Output: 1024.00000
Example 2:

Input: x = 2.10000, n = 3
Output: 9.26100
Example 3:

Input: x = 2.00000, n = -2
Output: 0.25000
Explanation: $2^{-2} = 1/2^2 = 1/4 = 0.25$

```python
class Solution:
    def myPow(self, x: float, n: int) -> float:
        if n==0:
            return 1
        if n<0:
            x = 1/x
            n = -n
        if n%2==0:
            return self.myPow(x*x, n//2)
        else:
            return x* self.myPow(x*x, n//2)
```

[9. Palindrome Number(6)]

Given an integer x, return true *if* x *is a* palindrome*,* and false otherwise.

Example 1:

Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
Example 2:

Input: x = -121

Output: false
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.
Example 3:

Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

```python
class Solution:
    def isPalindrome(self, x: int) -> bool:
        a = str(x)
        l = 0
        r = len(a)-1
        while l<r:
            if a[l]!=a[r]:
                return False
            l = l + 1
            r = r - 1
        return True
```

125. Valid Palindrome(7)

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.Given a string s, return true if it is a **palindrome**, or false otherwise.

Example 1:

Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
Example 2:

Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
Example 3:

Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric characters. Since an empty string reads the same forward and backward, it is a palindrome.

```python
class Solution:
    def isPalindrome(self, s: str) -> bool:
        l = 0
        r = len(s)-1
        while l<r:
            while l<r and not s[l].isalnum():
                l = l+1
            while l<r and not s[r].isalnum():
                r = r-1
            if s[l].lower() != s[r].lower():
                return False
            l = l+1
            r = r-1
        return True
```