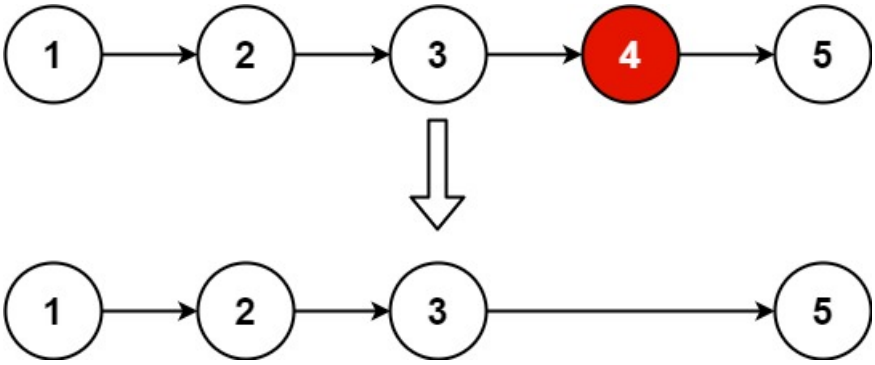
Prueba acceso Pickgram

Ejercicio 1

Dada la cabecera (**head**) de una lista enlazada (**Linked List**), elimina el elemento enésimo de la lista empezando por el final y retorna la cabecera.

Nota: No se requiere el uso de ningún paquete o libreria externa. Únicamente es necesario completar el código que viene al final de la explicación de este ejercicio. En el apartado ***código***

Ejemplo 1




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

Ejemplo 2

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

Ejemplo 3

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

Restricciones

- The number of nodes in the list is `sz`.
- `1 <= sz <= 30`
- `0 <= Node.val <= 100`
- `1 <= n <= sz`

Código

```
/**
 * Definition of a node in the linked list.
 * public class Node {
 *     int val;
 *     Node next;
 *     Node() {}
 *     Node(int val) { this.val = val; }
 *     Node(int val, Node next) { this.val = val; this.next = next; }
 * }
 */
class Problem {
    public Node removeNthNodeFromEnd(Node head, int n) {
    }
}
```

Instrucciones

1. Crear la clase Node definida en el comentario del código.
2. Copiar y crear la clase Problem.
3. Implementar el método removeNthNodeFromEnd. No es necesario crear el método public static void main. Pero lo puedes implementar si lo requieres para probar tu código.

Ejercicio 2

Dadas una serie de monedas con diferente valor y una cantidad total de dinero. Escribe una función para calcular la menor cantidad de monedas que son necesarias para alcanzar esa cantidad. Si esa cantidad de dinero no se puede recuperar con ninguna combinación de monedas, devuelve -1.

Puede suponer que tiene un número infinito de cada tipo de moneda.

Ejemplo 1

```
Input: coins = [1,2,5], amount = 11
Output: 3
Explicación: 11 = 5 + 5 + 1
```

Ejemplo 2

```
Input: coins = [2], amount = 3
Output: -1
```

Ejemplo 3

```
Input: coins = [1], amount = 0
Output: 0
```

Ejemplo 4

```
Input: coins = [1], amount = 1
Output: 1
```

Ejemplo 5

```
Input: coins = [1], amount = 2
Output: 2
```

Restricciones

- `1 <= coins.length <= 12`
- `1 <= coins[i] <= 231 - 1`
- `0 <= amount <= 104`

Código

```
class Problem {
    public int coins(int[] coins, int amount) {

    }
}
```

Instrucciones

1. Copiar y crear la clase Problem.
2. Implementar el método ***coins*** al cual se le pasa como argumentos un array de enteros que son las diferentes monedas disponibles y un amount que es el valor que queremos obtener con la suma de las monedas.

Ejercicio 3

¿Cómo diseñarías el ajedrez? ¿Qué clases y objetos usarías? ¿Qué métodos tendrían? No es necesario programarlo, una breve explicación es suficiente.