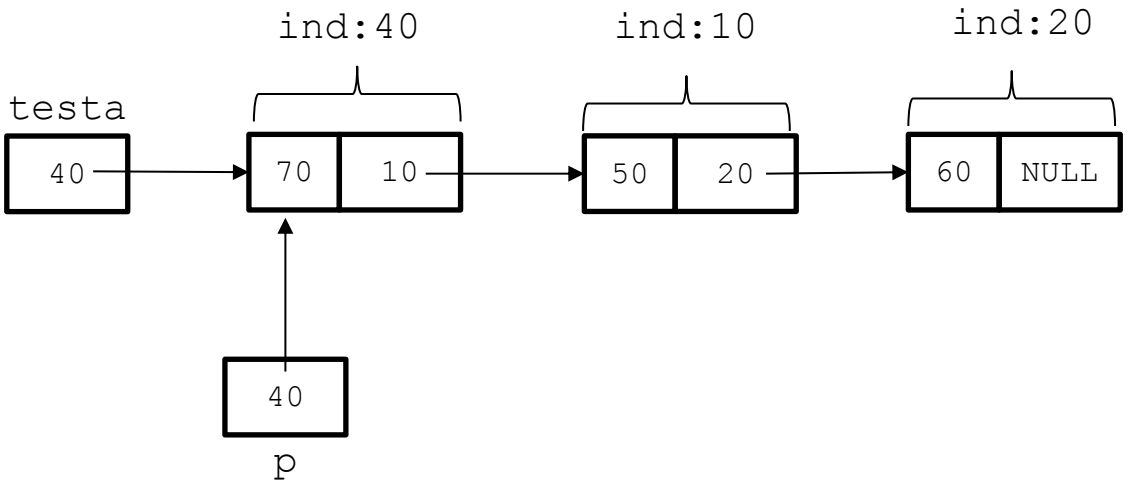
Esercizio 1

- Scrivere un funzione ricorsiva che stampa a video i valori di una lista di elementi con campo info di tipo intero. `stampa(elem* p);`

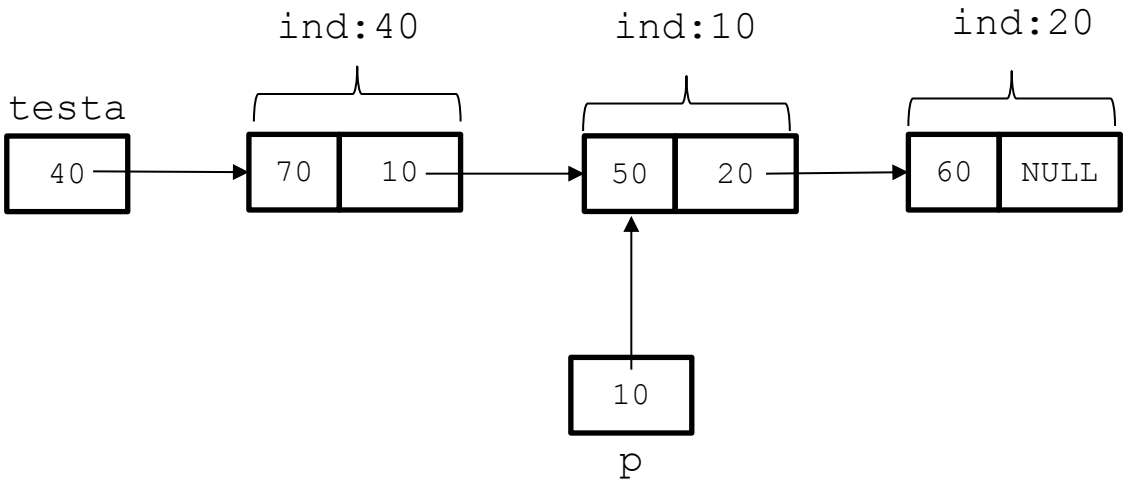
```
1° chiamata
stampa(p=40);
        il puntatore non è NULL, stampo 70 e chiamo stampa(p->pun)
```



Esercizio 1

- Scrivere un funzione ricorsiva che stampa a video i valori di una lista di elementi con campo info di tipo intero. `stampa(elem* p);`

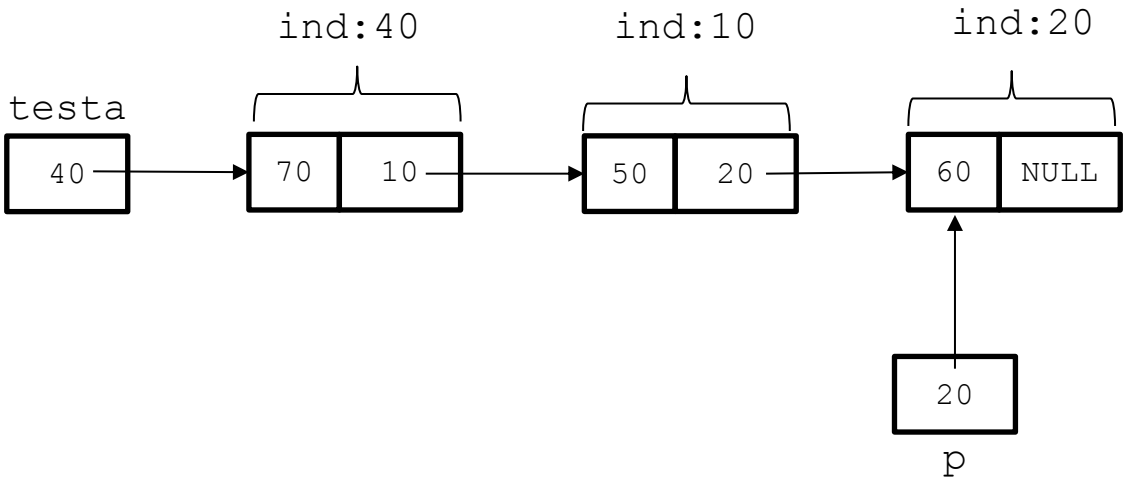
```
2° chiamata
stampa(p=10);
        il puntatore non è NULL, stampo 50 e chiamo stampa(p->pun)
```



Esercizio 1

- Scrivere un funzione ricorsiva che stampa a video i valori di una lista di elementi con campo info di tipo intero. `stampa(elem* p);`

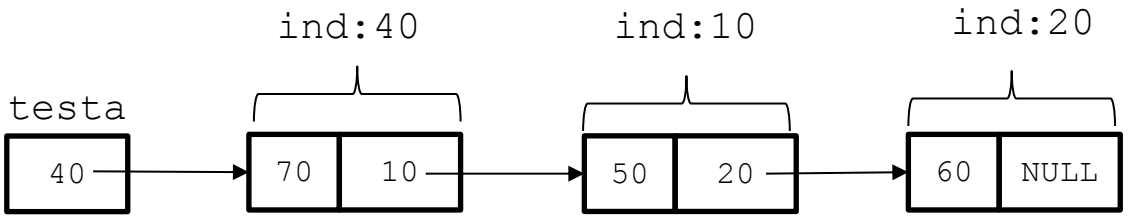
```
3° chiamata
stampa(p=20);
        il puntatore non è NULL, stampo 60 e chiamo stampa(p->pun)
```

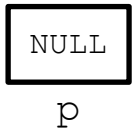


Esercizio 1

- Scrivere un funzione ricorsiva che stampa a video i valori di una lista di elementi con campo info di tipo intero. `stampa(elem* p);`

```
4° chiamata
stampa(p==NULL);
        Il puntatore è NULL,
        CASO BASE, non stampo nulla e non chiamo più la funzione stampa
```





Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

```
1° chiamata:
controlla(v,7)
      v[5] < v[6] ? No, chiamo controlla(v,6)
```

0	1	2	3	4	5	6
9	**7**	**3**	**0**	**-1**	**-8**	**-15**

Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

```
2° chiamata:
controlla(v,6)
      v[4] < v[5] ? No, chiamo controlla(v,5)
```

0	1	2	3	4	5
9	**7**	**3**	**0**	**-1**	**-8**

Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

```
3° chiamata:
controlla(v,5)
      v[3] < v[4] ? No, chiamo controlla(v,4)
```

0	1	2	3	4
9	**7**	**3**	**0**	**-1**

Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

```
4° chiamata:
controlla(v,4)
      v[2] < v[3] ? No, chiamo controlla(v,3)
```

0	1	2	3
9	**7**	**3**	**0**

Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

```
5° chiamata:
controlla(v,3)
      v[1] < v[2] ? No, chiamo controlla(v,2)
```

0	1	2
9	**7**	**3**

Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

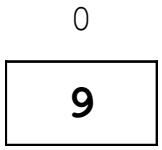
```
6° chiamata:
controlla(v,2)
      v[0] < v[1] ? No, chiamo controlla(v,1)
```

0	1
9	**7**

Esercizio 2

- Scrivere un funzione ricorsiva che controlla se un vettore è ordinato in maniera decrescente e restituisce true se lo è , false altrimenti.

```
7° chiamata:
controlla(v,1)
      E' un vettore di 1 elemento, è di per se'
      ordinato, non faccio controlli.
      CASO BASE: ritorno true
```



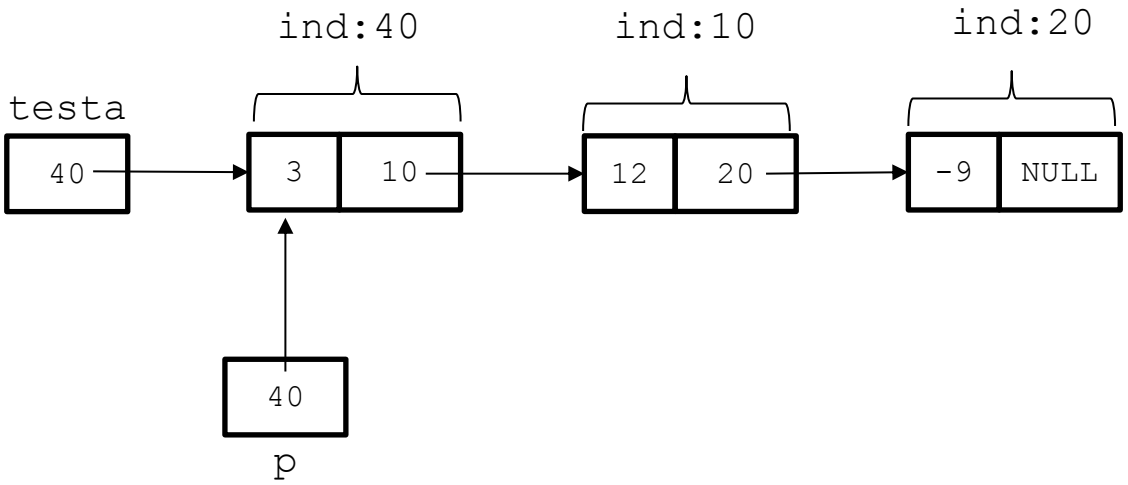

```
Cosa succedeva se uno dei controlli precedenti era
vero (cioè il penultimo elemento <= ultimo elemento)
Ritorno false
```

Esercizio 3

- Scrivere un funzione ricorsiva che conta il numero di elementi che hanno campo info minore di un numero intero n passato alla funzione

```
int conta(elem *p, int n)
```

```
1° chiamata
conta(p=40, n=4)
        il puntatore non è NULL,
3<4 ? Sì, ritorno 1+conta(p->pun,4)
```

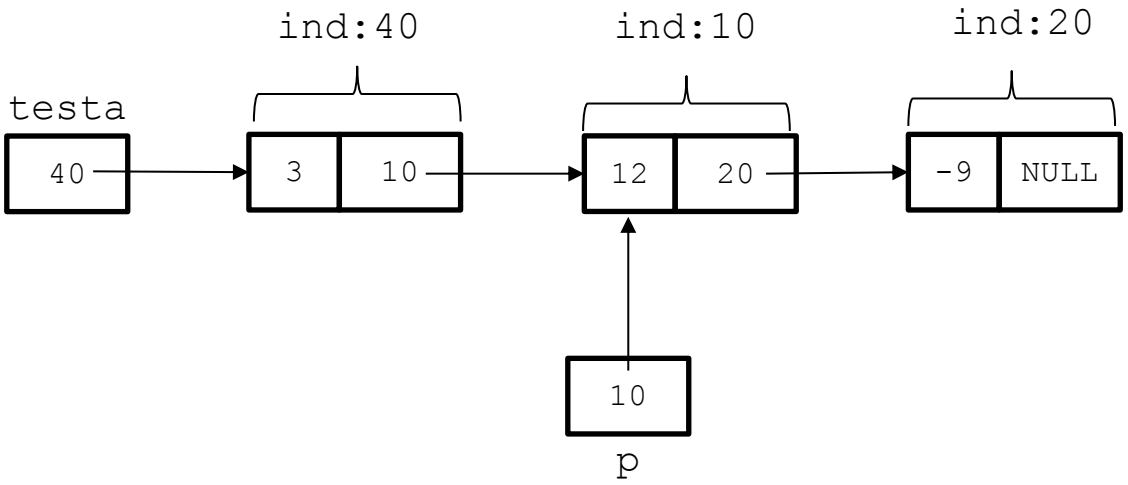


Esercizio 3

- Scrivere un funzione ricorsiva che conta il numero di elementi che hanno campo info minore di un numero intero n passato alla funzione

```
int conta(elem *p, int n)
```

```
2° chiamata
conta(p=10, n=4)
        il puntatore non è NULL,
12<4 ? No, ritorno 0+conta(p->pun,4)
```

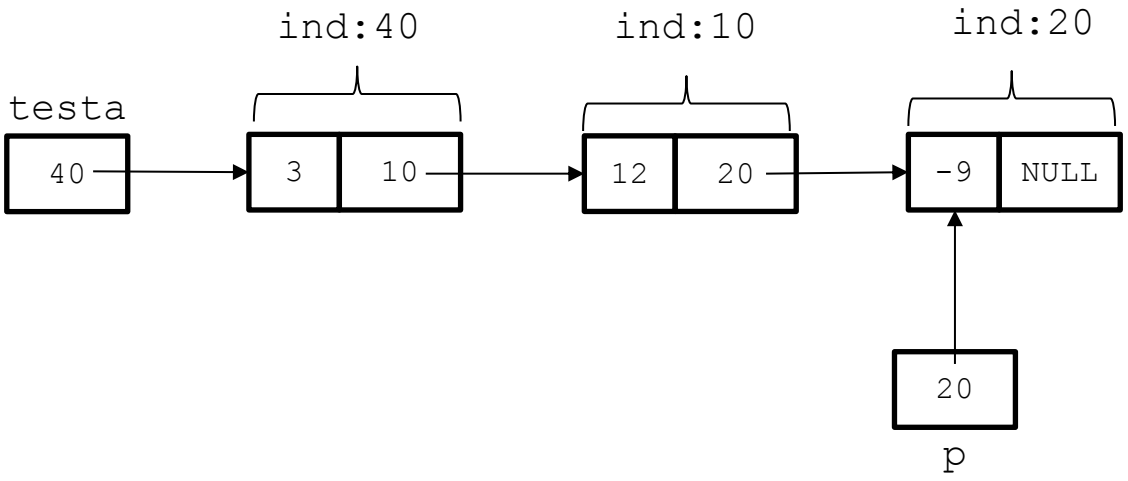


Esercizio 3

- Scrivere un funzione ricorsiva che conta il numero di elementi che hanno campo info minore di un numero intero n passato alla funzione

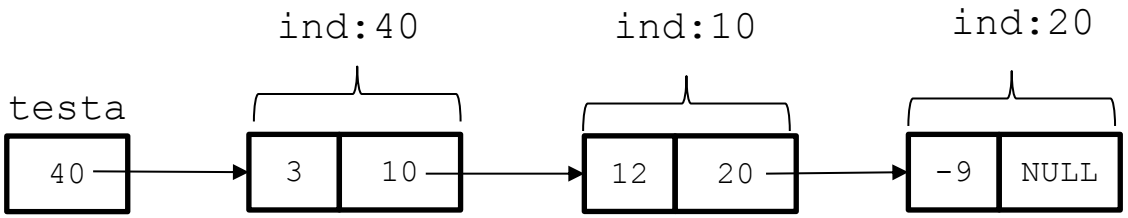
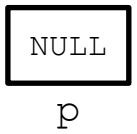
```
int conta(elem *p, int n)
```

```
3° chiamata
conta(p=20, n=4)
        il puntatore non è NULL,
-9<4 ? Sì, ritorno 1+conta(p->pun,4)
```



Esercizio 3

- Scrivere un funzione ricorsiva che conta il numero di elementi che hanno campo info minore di un numero intero n passato alla funzione

```
int conta(elem *p, int n)
```

```
4° chiamata
conta(p=NULL, n=4)
        il puntatore è NULL,
        CASO BASE: Ritorno 0, non ci sono elementi minori di n=4
```