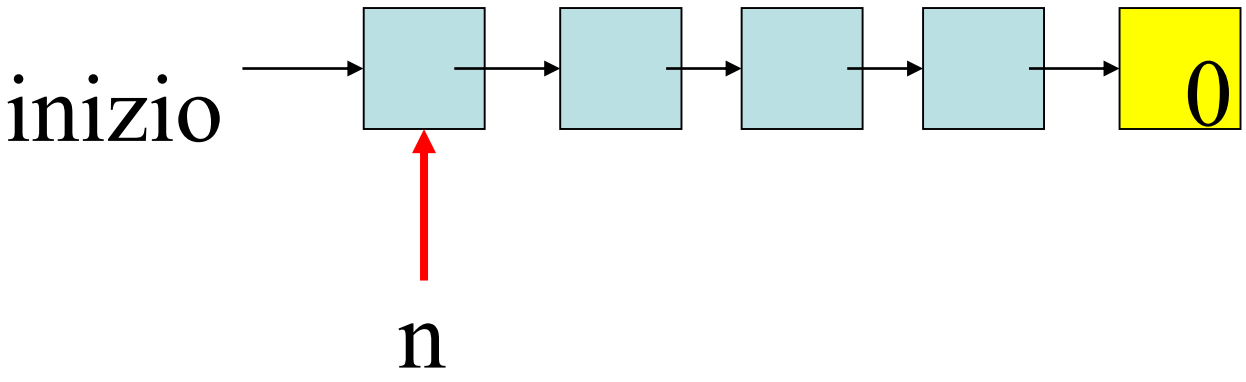
Ancora liste concatenate e ricorsione

usiamo quanto visto finora con un nuovo esercizio

eliminare l'ultimo nodo di una lista

I soluzione: quello che succede in pratica



n si deve fermare qui, quando n->next==0

=> deallocare questo nodo e restituire 0

I soluzione: la funzione riceve la lista originale e restituisce la nuova lista, tutto per valore

PRE=(L(n) è lista ben formata e non vuota, vL(n) =L(n))

nodo* delL1(nodo* n)

POST=(restituisce vL(n) – ultimo nodo)

PRE=(L(n) è lista ben formata e non vuota, vL(n)=L(n))

caso base

lista con un solo nodo

```
if(! n->next)
{delete n; return 0;}
```

POST=(restituisce vL(n) – ultimo nodo)

caso ricorsivo: lista con almeno 2 nodi:

si deve:

1) fare l'invocazione ricorsiva sul resto della lista e
2) appendere il risultato dell'invocazione al nodo corrente e restituirlo

```
n->next=delL1( n->next);
return n;
```

mettendo tutto insieme

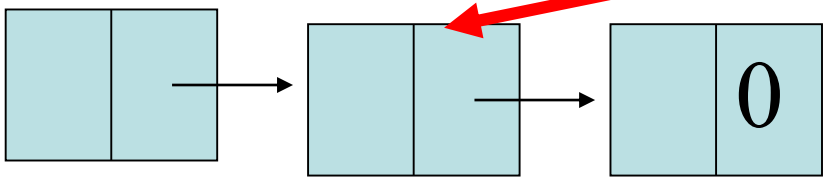
```
nodo * delL1(nodo *n)
{
   if( ! n→next )
        {delete n; return 0;}
    n→next=delL1(n→next);
    return n;
}
```

invocazione:
if(inizio) inizio=delL1(inizio);

operazioni inutili ?
Si
è possibile evitarlo?
Si, ma può costare in generalità

per fare solo l'operazione che serve

dobbiamo fermare la ricorsione al penultimo nodo



poco generale
BRUTTA !!

<u>caso base:</u>

n→next→next == 0

funziona solo per liste con almeno 2 nodi !!!

II soluzione

PRE=(L(n) è ben formata con almeno 2 nodi)

```
void  delL2(nodo *n)
{if( ! n→next→next )
        {delete n→next;  n→next=0; }
  else
         delL2(n→next);
}
```

POST=(L(n)=vL(n)-ultimo nodo)

primo nodo non cambia

III soluzione: col passaggio per riferimento

passando n->next per riferimento, arriviamo all'ultimo nodo avendo un alias del campo next del nodo precedente

e se il nodo precedente non c'è, allora abbiamo un alias della variabile che punta all'inizio della lista

PRE=(L(n) è ben formata non vuota)

```
void delL3(nodo *& n)
{
  if(!n→next)
        {delete n; n=0; }
  else
        delL3(n →next);
}
```

POST=(L(n)=vL(n) – ultimo nodo)

invocazione: if(inizio) delL3(inizio);

ricorsione e passaggio per riferimento:

```
void f(… int &x ….)
{
….f(…x…)…
}
```

tutte le invocazioni di f condividono la variabile x e quindi ogni modifica di x si ripercuote su tutte le invocazioni

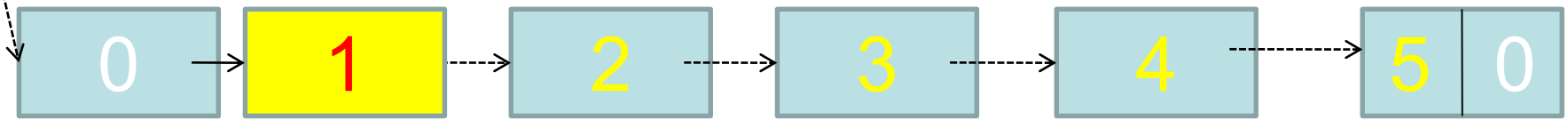
ma negli esercizi visti finora era:

```
void f(nodo * & n….)
{
….f( n->next…)
}
```

chiamante e chiamato hanno in comune il campo next del nodo puntato dal chiamante

esercizio: inserire un nodo in posizione k=0,1,..

per esempio se k=1



ma se k>5 non si può fare e Q resta uguale

conviene introdurre la seguente notazione:

data L(Q) lista ben formata

Lk(Q)= lista che consiste dei primi k nodi di L, cioè dal nodo 0 al k-1
attenzione ➔ L0(Q) = lista vuota

il nodo finale di Lk(Q) è quello in posizione k-1, quindi se L(Q) = Lk(Q)@R la nuova lista è Lk(Q)@nodo(..)@R

-I soluzione:
una funzione a cui diamo la lista e k e ci restituisce la lista modificata

caso base, che sappiamo risolvere facilmente: il nuovo nodo va inserito al primo posto della lista, k=0

ricorsione se k>0

PRE=(L(Q) ben formata, k>=0, vL(Q)=L(Q))

```
nodo* ins1(nodo*Q, int k, int c)
{
  if(!k) return new nodo(c, Q);
  if(!Q) return 0;
  Q->next=ins1(Q->next,k-1,c);
   return Q;
}
```

POST=(se vL(Q)=vLk(Q)@R, restituisce vLk(Q)@nodo(c)@R, altrimenti vL(Q))

come al solito fa operazioni inutili al ritorno

-II soluzione : vogliamo fare solo le operazioni necessarie

la ricorsione si deve fermare al nodo che precede quello che dobbiamo aggiungere: il nodo k-1

non va con k=0 ! Non c'è nodo che precede

base: k=1

invocazione ricorsiva: k>1

PRE=(L(Q) ben formata e non vuota, k>0, vL(Q) =L(Q))

```
void ins2(nodo*Q, int k, int c)
{
   if(k==1)
       Q->next=new nodo(c,Q->next);
   else
       if(Q->next)
           ins2(Q->next, k-1,c);
}
```

POST=(se vL(Q)=vLk(Q)@R, L(Q)=vLk(Q)@nodo(c)@R, altrimenti L(Q)=vL(Q))

III soluzione con passaggio di Q per riferimento

PRE=(L(Q) ben formata, k>=0, vL(Q)=L(Q))

```
void ins3(nodo*& Q, int k, int c)
{
   if(k==0)
       Q=new nodo(c,Q);
   else
       if(Q)
           ins3(Q->next, k-1, c);
}
```

POST=(se vL(Q)=vLk(Q)@R, L(Q)= vLk(Q)@nodo(c)@R, altrimenti L(Q)=vL(Q))