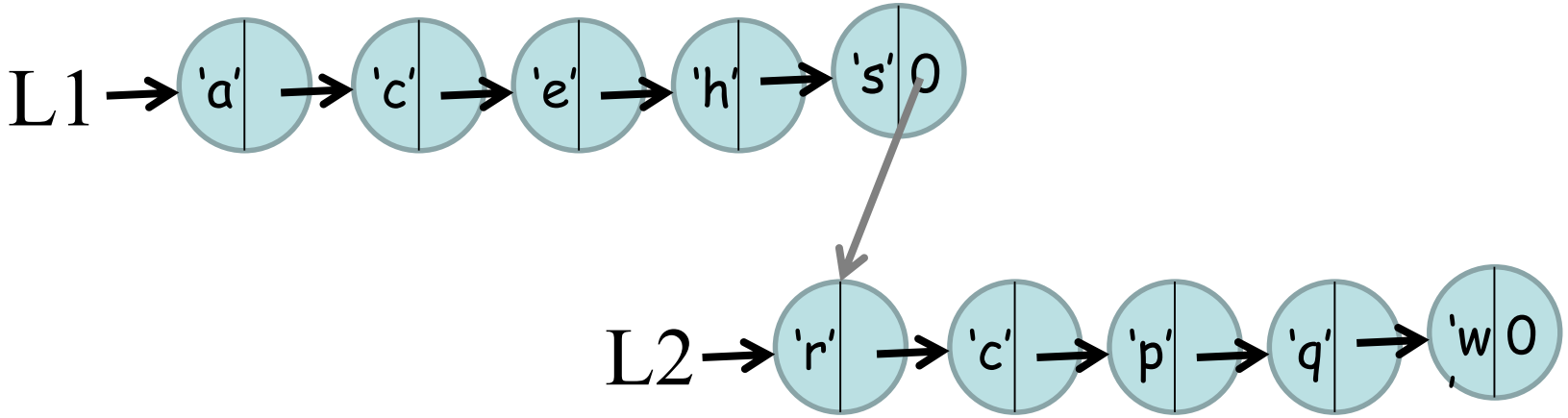
esempi di funzioni risorsive su liste concatenate

concatenazione di liste:



potrebbe essere che L1 e/o L2 siano vuote

usiamo la notazione: Lista(L1) @ Lista(L2)

funzione concatenazione: usiamo metodo 1

PRE=(Lista(L1) e Lista(L2) ben formate)

```
nodo* conc(nodo* L1, nodo* L2)
{
  if(!L1) return L2;
  L1->next=conc(L1->next, L2);
  return L1;
}
```

POST=(restituisce Lista(L1)@Lista(L2))

metodo 2: dobbiamo fermarci sull'ultimo nodo di L1 che deve quindi non essere vuota

PRE=(L(L1) e L(L2) ben formate e la prima non vuota, vL(L1)=L(L1) e vL(L2)=L2)

```
void conc(nodo* L1, nodo* L2)
{
  if(!L1->next)
       L1->next=L2;
  else
     conc(L1->next, L2);
}
```

POST=(L(L1)=vL(L1)@vL(L2))

metodo 3: passaggio per riferimento

PRE=(L(L1) e L(L2) ben formate e la prima non vuota, vL(L1)=L(L1) e vL(L2)=L2)

```
void conc(nodo* & L1, nodo* L2)
{
  if(!L1)
       L1=L2;
  else
     conc(L1->next, L2);
}
```

POST=(L(L1)=vL(L1)@vL(L2))

Consideriamo il problema dell'Esercizio 2 della settimana scorsa
eliminare da una lista concatenata tutti i nodi con un certo info=z

ma ora non vogliamo deallocare i nodi tolti dalla lista, ma con essi costruiamo una nuova lista che va restituita
in qualche modo. Come facciamo con del1?

nodo* del1(nodo*L, int z) il return è già occupato !
come restituiamo la seconda lista?

Metodo 1

```
struct doppiaL{nodo*L,*S; doppiaL(nodo*a=0,nodo*b=0){L=a; S=b;}};
doppiaL del1_ret(nodo*L,int z) //PRE=(L(L) ben formata e vL(L)=L(L))
{
 if(L)
   {
    if(L->info==z)
            {
             doppiaL q = del1_ret(L->next,z);
             L->next=q.S;
             q.S=L;
             return q;
            }
    else
            {
             doppiaL q =del1_ret(L->next,z);
             L->next=q.L;
             q.L=L;
             return q;
            }
   }
 return doppiaL();
}POST=(restituisce doppiaL D con D.L=vL(L) meno i nodi con info=z e D.S è composta dai nodi con info=z)
```

Metodo 2

PRE=(L(L) benformata e non vuota e L->info!=z, vL(L)=v(L))

```
nodo* del2_ret(nodo*L, int z)
{
 if(L->next)
   {
    if(L->next->info==z)
            {
             nodo*y=L->next;
             L->next=L->next->next;
             y->next=del2_ret(L,z);
             return y;
            }
    else
            {
             return del2_ret(L->next,z);
            }
   }
 else
   return 0;
}
```

POST=(L(L)=vL(L) meno i nodi con info=z la cui lista è restituita col return)

metodo 3

```
//PRE=(L(L) ben formata e vL(L)=L(L))

nodo* del3_ret(nodo*&L,int z)
{
  if(L)
    {
      if(L->info==z)
                {
                  nodo*x=L;
                  L=L->next;
                  x->next=del3_ret(L,z);
                  return x;
                }
      else
                return del3_ret(L->next,z);
    }
  else
    return 0;
}
POST=(L(L)=vL(L) meno i nodi con info=z la cui lista è restituita col return)
```