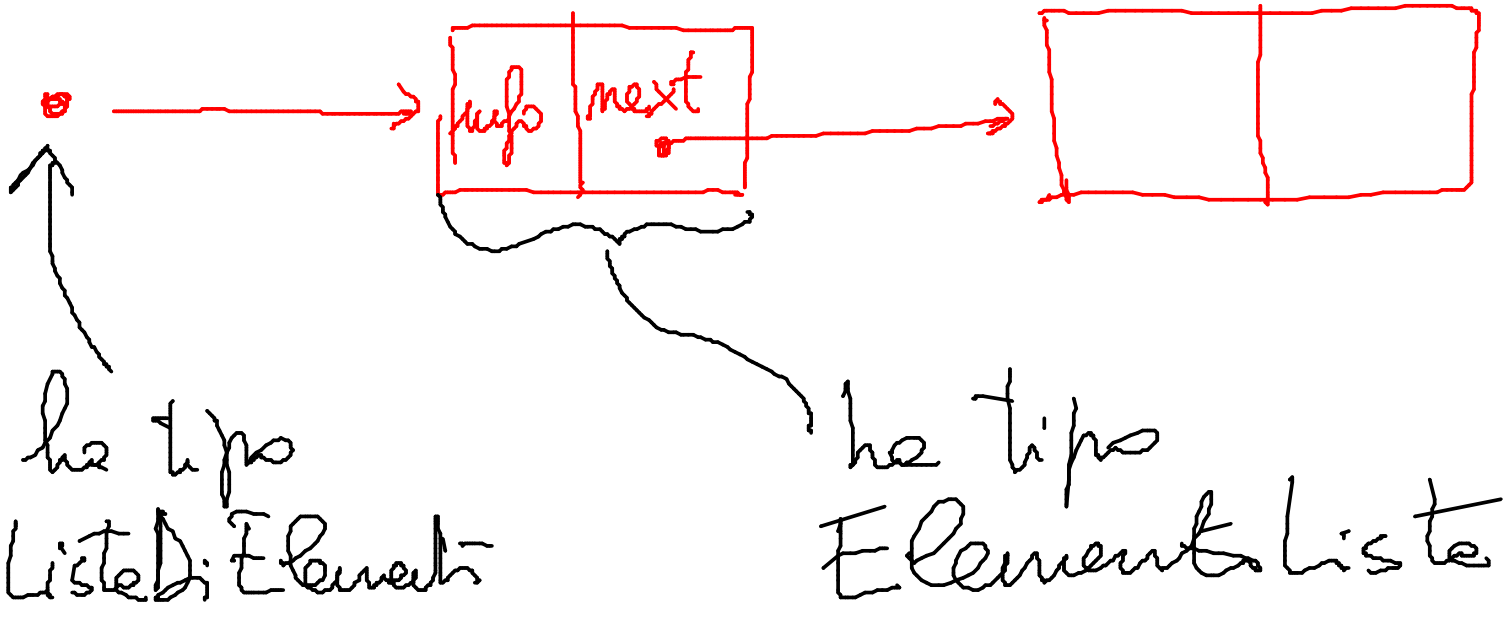
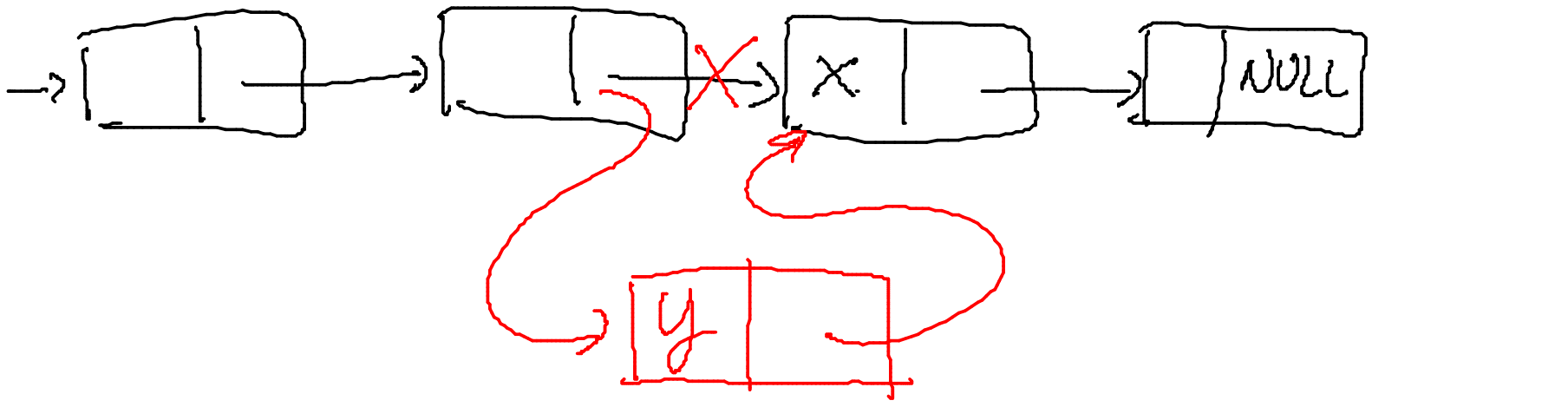
```
struct el { int info;
            struct el* next;}
typedef struct el ElementoLista;
typedef ElementoLista * ListaDiElementi;
```

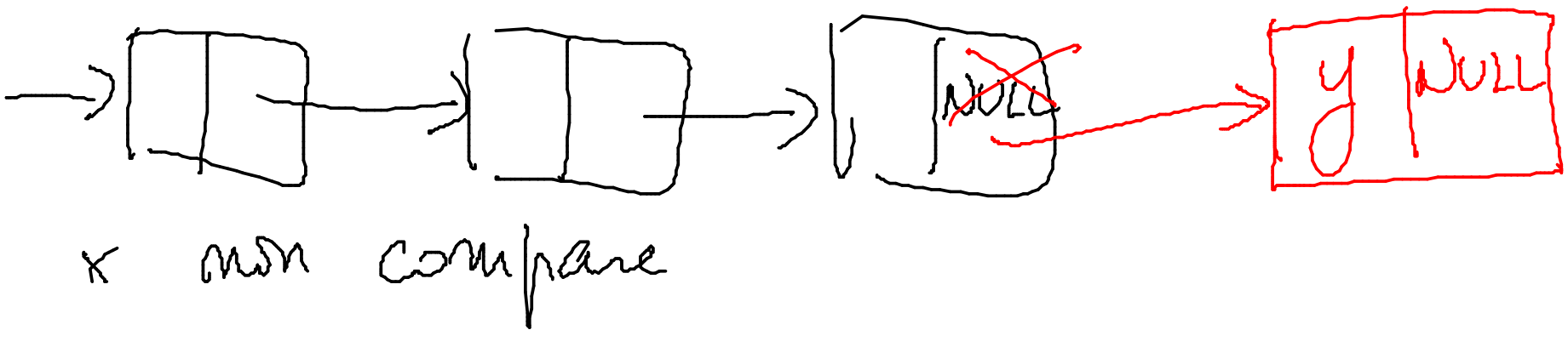
Tipo

Nome



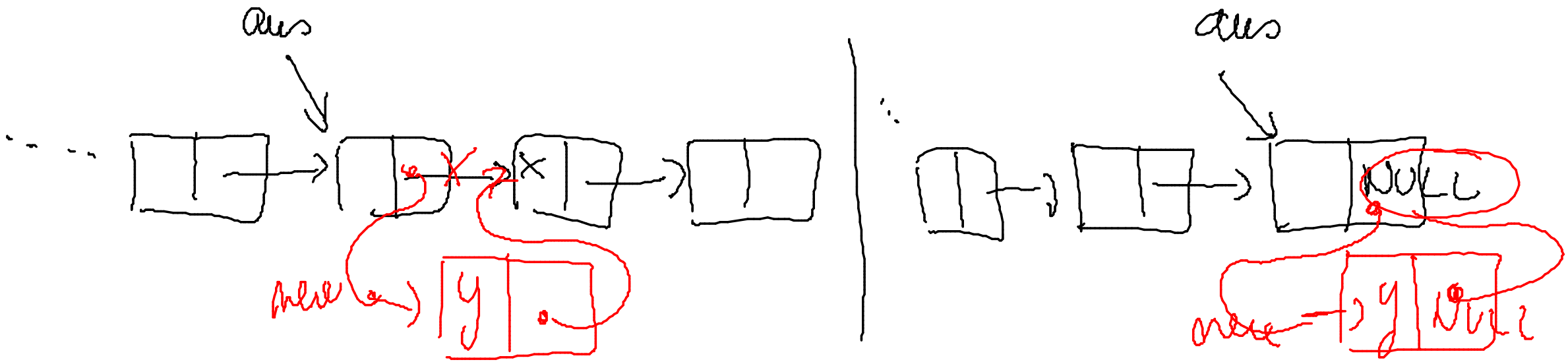
Date una lista inserire un elemento che contiene
y prima del primo elemento che contiene x.
Se x non compare nella lista inserire y alla fine





```
void ins ( ListeDiElement * l, int y, int x)
{ ListeDiElement new = malloc( ... ); new -> info = y;
  if (*l == NULL) { *l = new; new -> next = NULL; }
  else if (*l -> info == x) { new -> next = *l; *l = new; }
       else { ListeDiElem prec = *l; ListeDiElement corr = *l -> next;
              int trovato = 0;
              while ( corr != NULL && ! trovato)
                if (corr -> info == x) trovato = 1;
                else { prec = corr; corr = corr -> next; }
              prec -> next = new;
              new -> next = corr;
       }
}
```

```
{ listaInElement aus = *l; int trovato = 0;
while (aus->next != NULL && !trovato)
  if (aus->next->info == x) trovato = 1;
     else aus = aus->next;
new->next = aus->next;
aus->next = new;
}
```

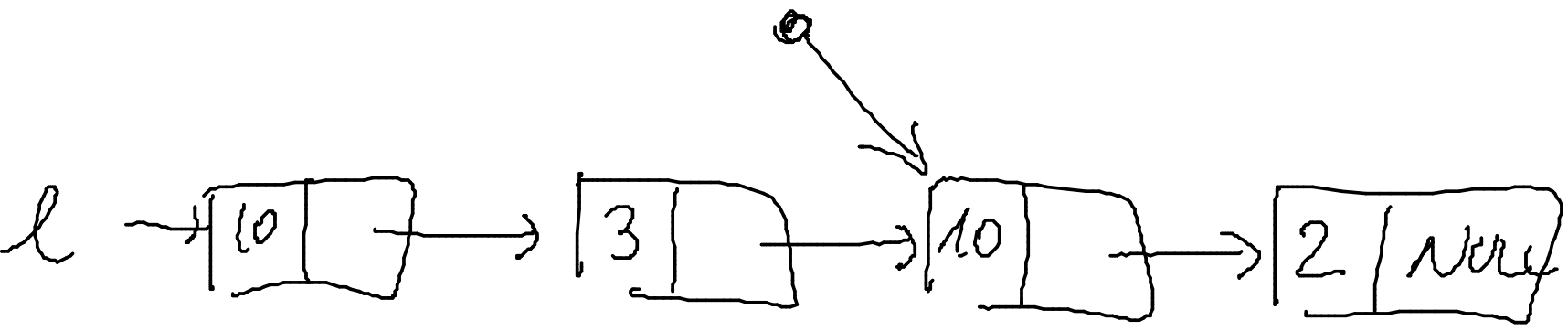


Cercare il massimo di una lista di interi non vuota.

```
int max (ListaDiElement l)
{
  int maxel = l->info;
  ListaDiElement aux = l->next;
  while(aux != NULL)
  {
    if (aux->info > maxel) maxel = aux->info;
    aux = aux->next;
  }

return maxel;
}
```

funzione che restituisce il puntatore all'ultima occorrenza del massimo

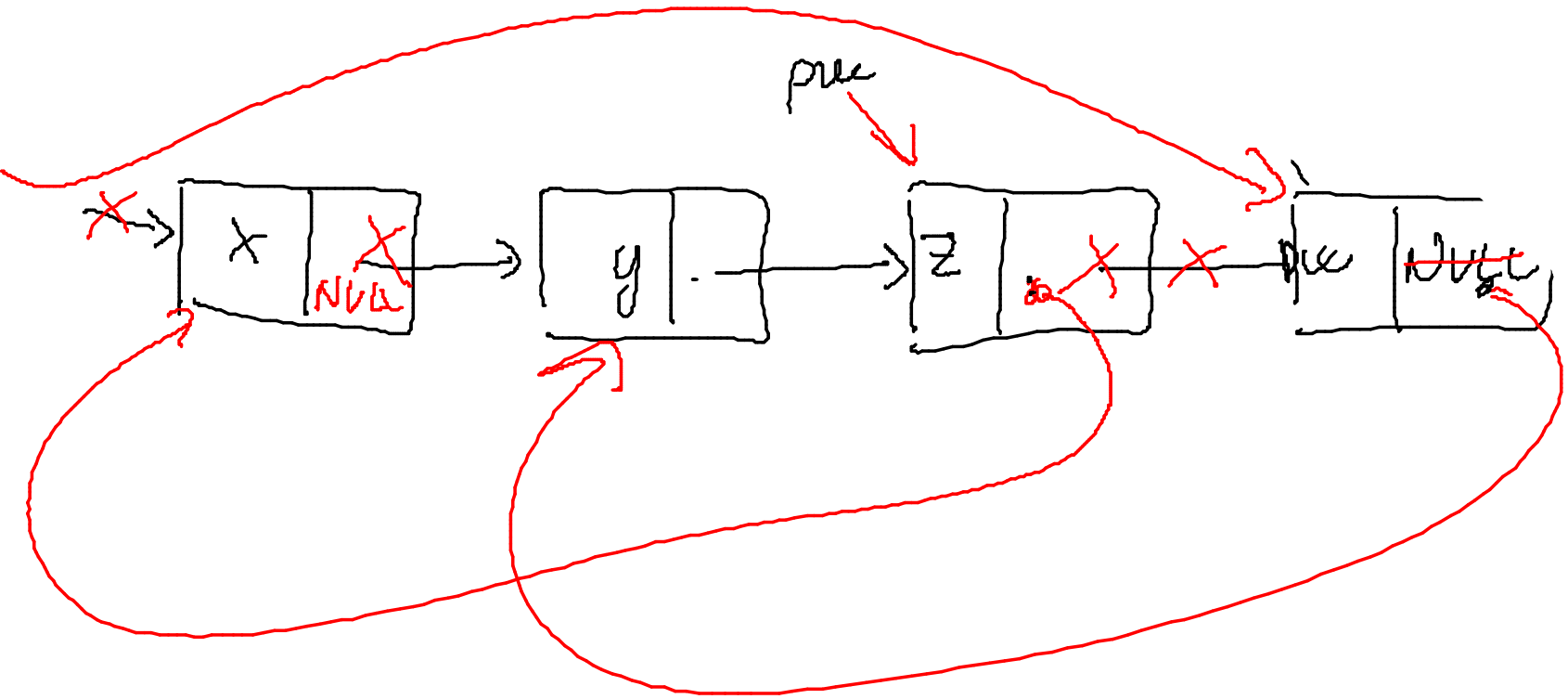


ListeDiElementi pmax (ListeDiElementi l)

```
listeDiElement pmax (listeDiElement l)
{ int maxel = l->info; listeDiElement pmaxel = l;
  listeDiElement aus = l->next;
  while ( aus != NULL)
  { if ( aus->info >= maxel)
    { maxel = aus->info; pmaxel = aus;}
    aus = aus->next;
  }
  return pmaxel;
}
```

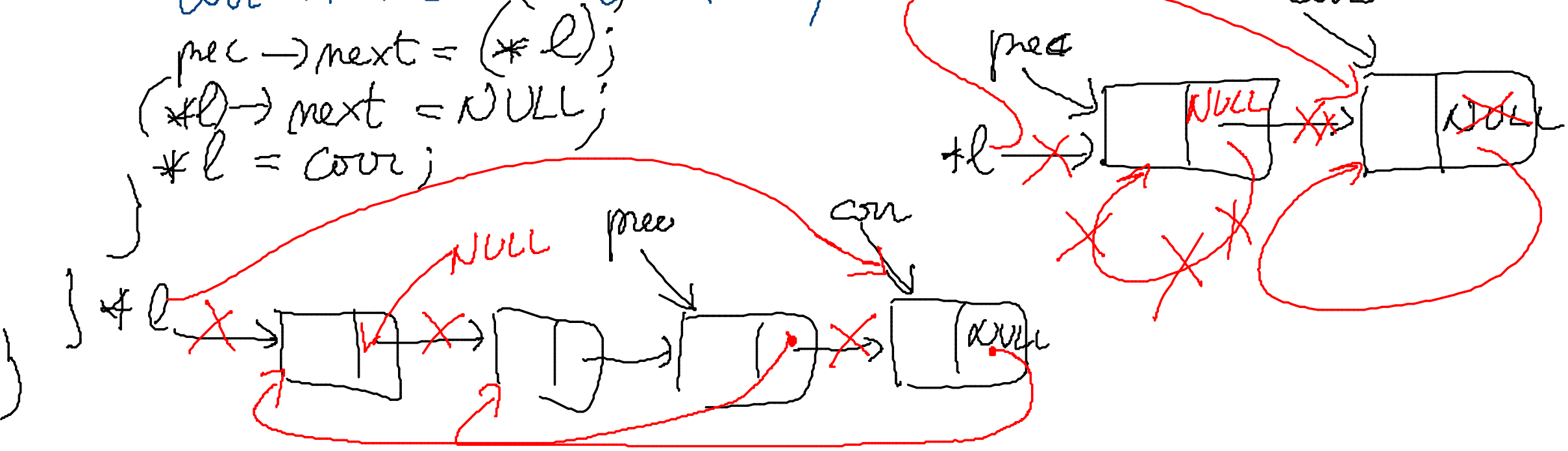
pmaxel->info

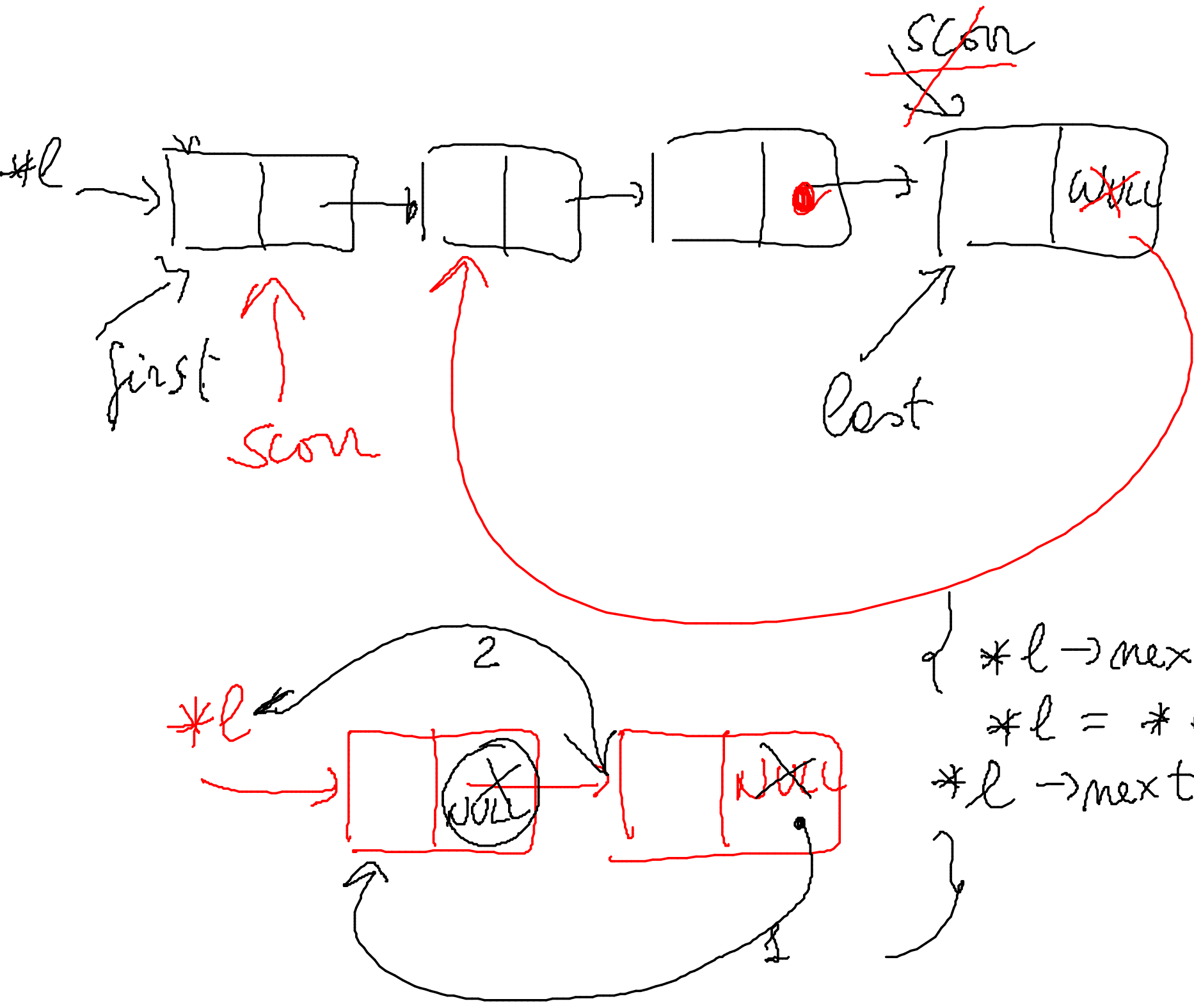
Scambiare il primo elemento di una lista con l'ultimo senza fare assegnamenti sui campi info




```
curr->next -> *l -> next
*l->next = NULL
pre->next = *l
*l = curr
```

```
void scambia (ListeDiElement * l)
{ if ((*l) != NULL)
  { if ((*l)->next != NULL)
    { ListeDiElement prec = *l; ListeDiElement corr = (*l)->next;
      while ( corr -> next != NULL) { prec = corr ; corr = corr -> next ; }
      corr -> next = (*l) -> next;
      prec -> next = (*l);
      (*l) -> next = NULL;
      *l = corr;
    }
  }
}
```






```
{ *l->next->next = *l;
  *l = *l->next;
*l->next->next = NULL;
```