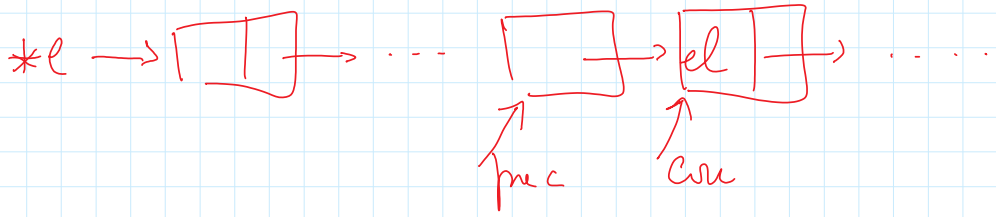
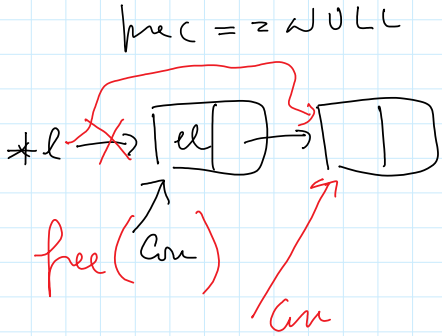
Scrivere una procedura C che cancella da una lista tutte le occorrenze di un elemento el.

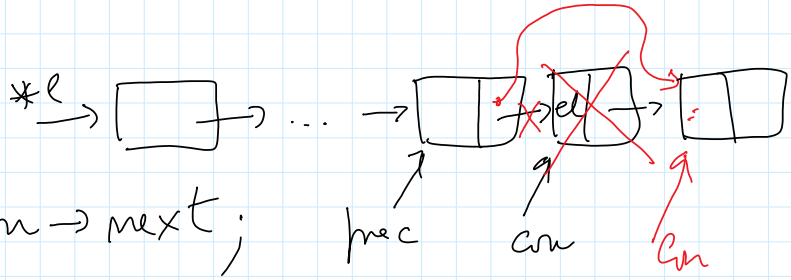
```
void cancella (int el, ElementoDiLista** l)
{
```




```
ElementoDiLista * prec = NULL;
ElementoDiLista * corr = *l;
while (corr != NULL)
  if (corr->info == el)
    if (prec == NULL)
    {
      *l = *l->next;
      free(corr);
      corr = *l;
    }
    else
    {
      prec->next = corr->next;
      free (corr);
      corr = prec->next;
    }
  else
```

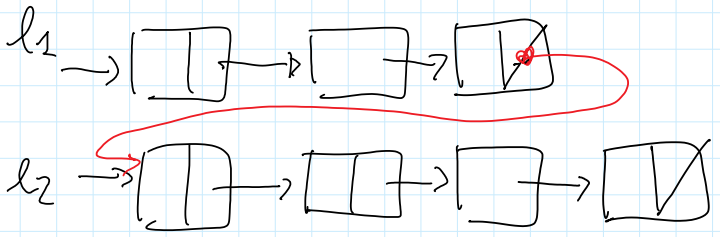
= → `*l = corr->next;` ok

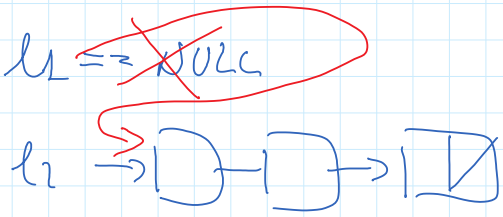




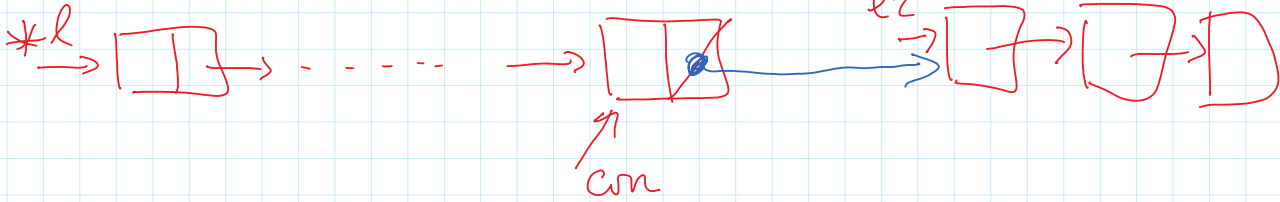
```
    {
      prec = corr;
      corr = corr -> next;
    }
}
```

Scrivere una procedura C che date due liste l1 e l2 concatena la lista l2 alla fine della lista l1.



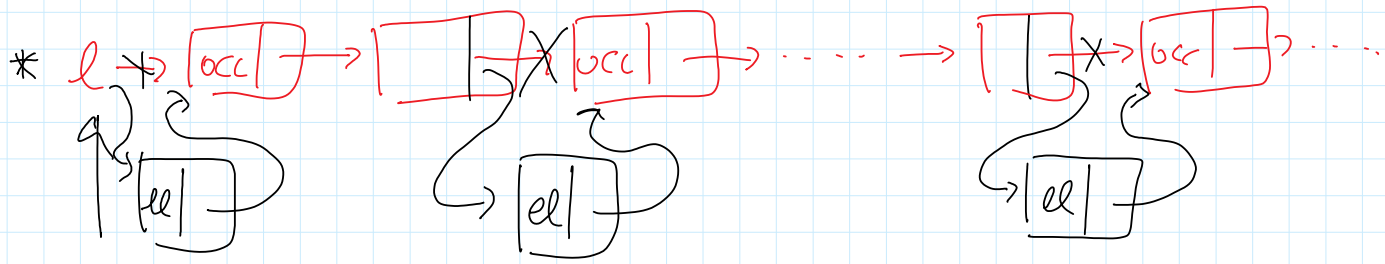



```
void conc ( ElementoDiLista ** l1,
            ElementoDiLista *  l2)
{ if (*l1 == NULL) *l1 = l2;
  else
  { ElementoDiLista * corr = *l1;
    while (corr->next != NULL)
      corr = corr->next;
```

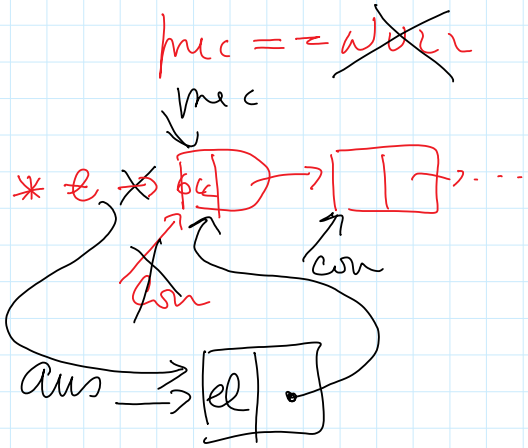



```
    corr->next = l2;
  }
```

Scrivere una procedura C che aggiunge, in una lista, prima di tutte le occorrenze di un elemento occ, l'elemento el.



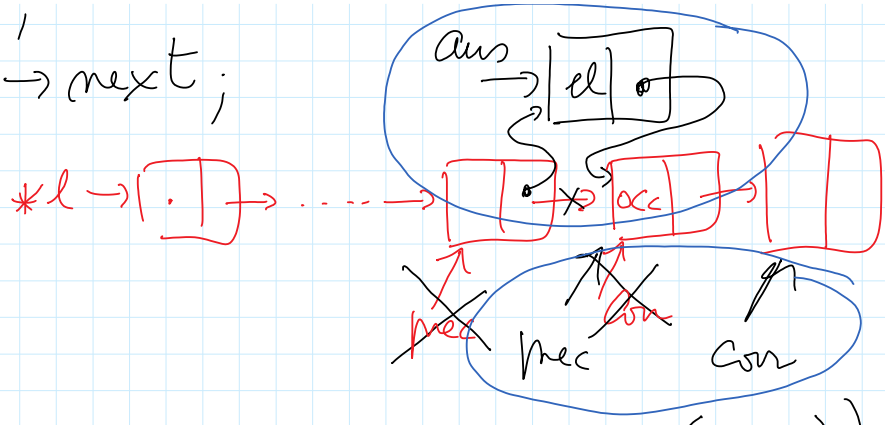

```
void ins(int el, int occ, ElementoDiLista **l)
{ ElementoDiLista * prec = NULL;
  ElementoDiLista * corr = *l;
  while (corr != NULL)
    if (corr->info == occ)
      if (prec == NULL)
```



ElementoDiLista

```
{ ElementoDiLista * aus = malloc(sizeof(EDL));
  aus->info = el;
  aus->next = corr;
  *l = aus;
  prec = corr;
  corr = corr->next;
```

```
        prec = corr;
        corr = corr -> next;
    }
    else
    {
```



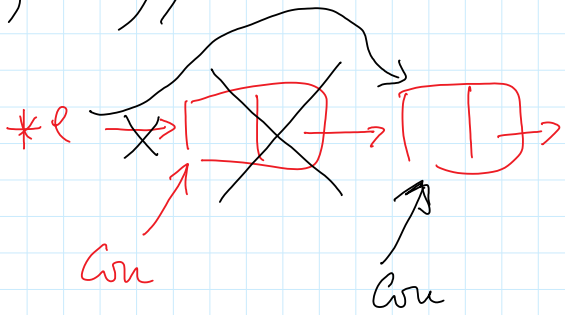

```
        ElementoDiListe * aus = malloc(sizeof(EDL));
        aus -> info = el;
        aus -> next = corr;
        prec -> next = aus;
        prec = corr;
        corr = corr -> next;
    }
}
else
{
    prec = corr;
    corr = corr -> next;
}
}
```

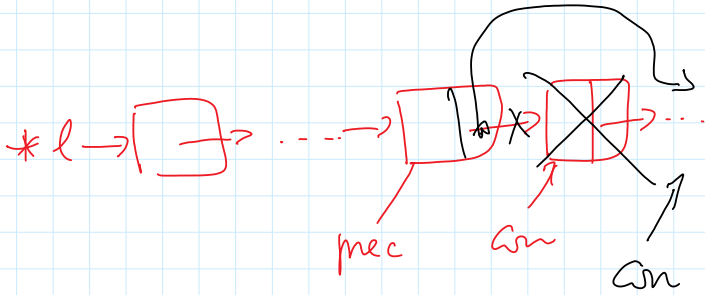
Scrivere una procedura C che, date due liste, l1 e l2, cancella da l1 tutti gli elementi che non compaiono in l2.

member

```
int member (int el, ElementoDiLista * l)
{ int trovato = 0;
  ElementoDiLista * corr = l;
  while (corr != NULL && !trovato)
     if (corr -> info == el) trovato = 1;
     else corr = corr -> next;
  return trovato;
}
```

```
Void conc ( ElementoDiLista ** l1 ;
            ElementoDiLista *  l2 )
{ ElementoDiLista * prec = NULL;
  ElementoDiLista * corr = *l1;
  while ( corr != NULL)
    if (! member (corr -> info , l2))
      if (prec == NULL)
      {
        *l = *l -> next;
        free (corr);
        corr = *l;
      }
      else
      { prec -> next = corr -> next;
        free (corr);
        corr = prec -> next;
      }
  else
  { prec = corr;
```

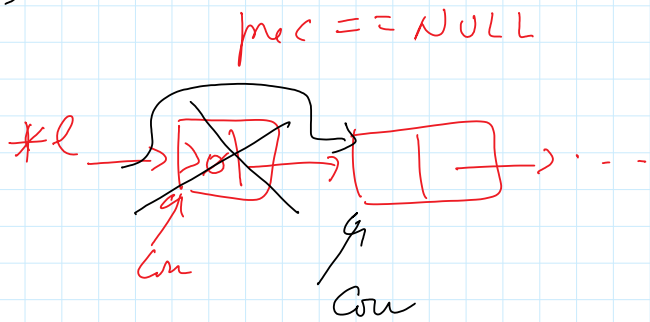


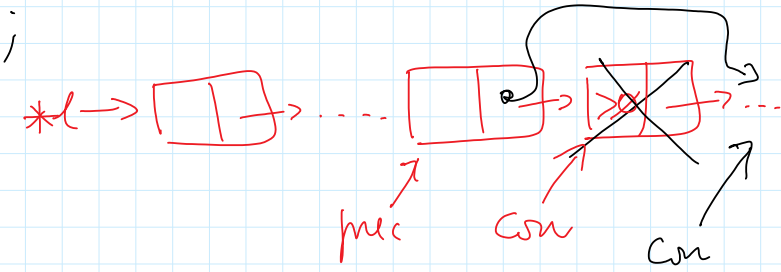


```
    {  prec = curr;
       curr = curr -> next;
    }
}
```

Scrivere una procedura C che, dato un intero positivo n, cancella da una lista i primi n elementi maggiori di 0. Se nella lista ci sono meno di n elementi > 0 vengono cancellati tutti.

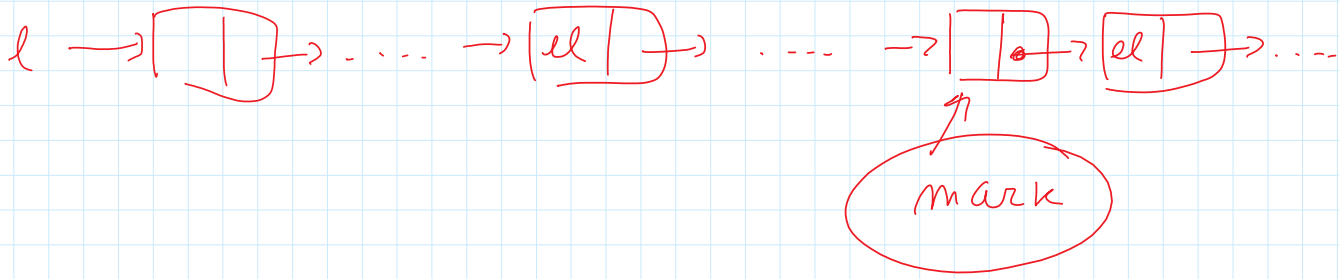
```
void canc (int n, ElementoDiLista ** l)
{ ElementoDiLista * prec = NULL;
  ElementoDiLista * corr = *l;
  while (n > 0 && corr != NULL)
    if (corr -> info > 0)
      if (prec == NULL)
      {
        n = n-1;
        *l = *l -> next;
        free(corr);
        corr = *l;
      }
      else
      { prec -> next = corr -> next;
```





```
        {  prec->next = curr->next;
           free(curr);
           curr = prec->next;
        }
    else
    {   prec = curr;
        curr = curr->next;
    }
}
```

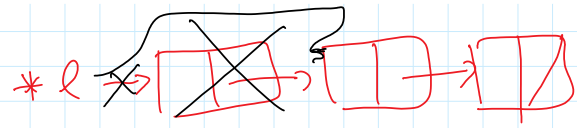
Scrivere una procedura C che, dete una lista e un intero el, cancelle l'ultima occorrenza di el (se c'è).




```
void canc (int el, ElementoDiLista **l)
{ ElementoDiLista *mark = NULL;
  ElementoDiLista *prec = NULL;
  ElementoDiLista *corr = *l;
  int trovato = 0;
  while (corr != NULL)
    if (corr->info == el)
    { trovato = 1;
      mark = prec;
      prec = corr;
      corr = corr->next;
    }
    else
    { prec = corr;
      corr = corr->next;
    }
  if (trovato)
    if (mark == NULL)
```

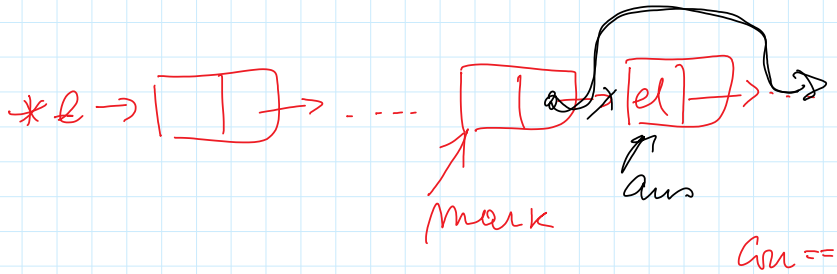
tutte volte quante sono le occorrenze di el.

mark == NULL
prec == NULL



curr == NULL

```
{
    ElementoDiListe * aus = *l;
    *l = *l -> next;
    free (aus);
}
```



curr == NULL

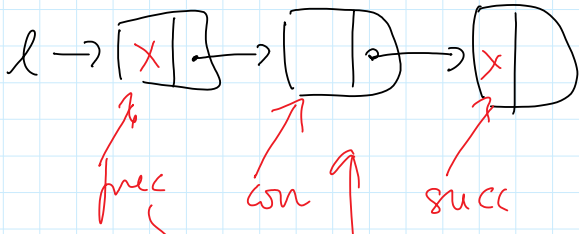
```
else
{
    ElementoDiListe * aus = mark -> next;
    mark -> next = aus -> next;
    free (aus);
}
}
```

```
{ if (curr -> info == el)
    { trovato = 1;
      mark = prec;
    }
  prec = curr;
  curr = curr -> info;
}
```

}

Scrivere una procedura C che cancella, se c'è, il primo elemento preceduto e seguito da due elementi uguali.

- Per fare qualcosa nella lista devo avere almeno 3 elementi.




```
if (l != NULL)
  if (l -> next != NULL)
    if (l -> next -> next != NULL)
    { faccio qualcosa! }
```

prec -> info == succ -> info

Condizione