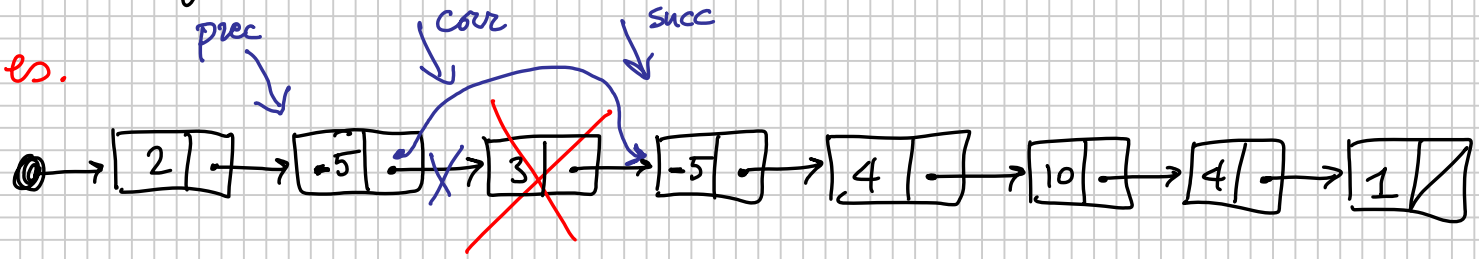
ESERCIZI su LISTE in C

Titolo nota 26/11/2015

Scrivere una procedura che, data una lista di interi, cancella il primo elemento che è preceduto e seguito da 2 elementi uguali e lascia la lista inalterata se tale elemento non esiste.

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

es.



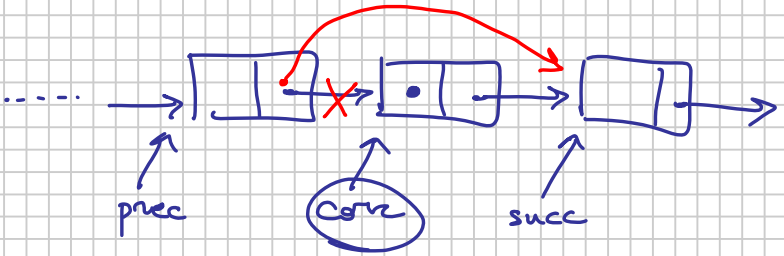

```
void cancella(ListaDiElementi lista)
```

```
void cancella (ListaDiElementi lista)
{   if (lista != NULL)
      { if (lista -> next) != NULL)
          { ListaDiElementi prec, corr, succ; int trovato = 0;
            prec = lista;
            corr = prec -> next;
            succ = corr -> next;
            while ( succ != NULL && !trovato)
              if ( prec -> info == succ -> info)
                 { trovato = 1;
                   prec -> next = succ;
                   free (corr);
                 }
              else
                 {  prec = corr;
                    corr = succ;
                    succ = corr -> next;
                 }
          }
      }
}
```

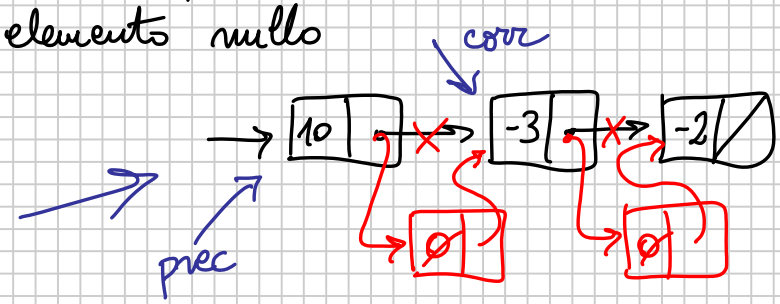
prec = prec -> next;

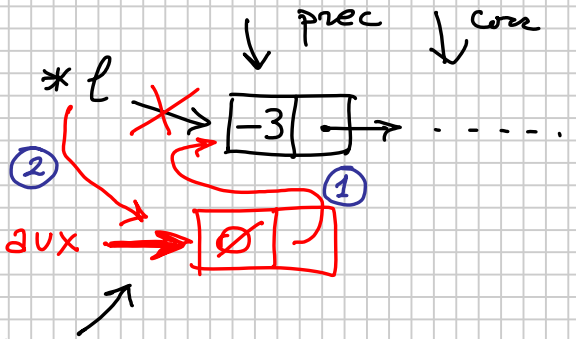
prec = corr;
corr = prec -> next;
succ = corr -> next;

(prec -> next = corr -> next;)



Scrivere una procedura che, data una lista di interi, antepone ad ogni elemento negativo un elemento nullo

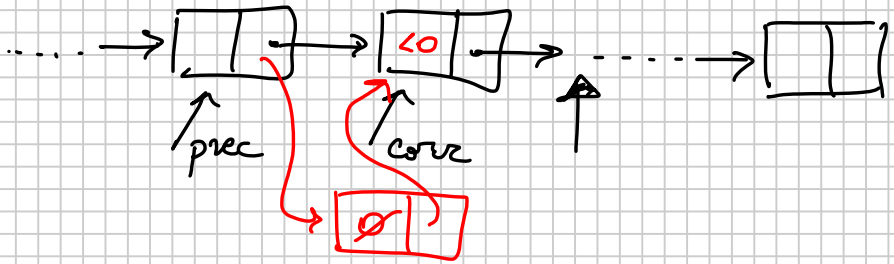





```
void pippo (ListaDiElementi *lista) { ListaDiElementi prec, corr;
    if (*lista != NULL)
      { if ((*lista)→info < ∅) { ListaDiElementi aux = malloc (sizeof (ElementoDiLista));
                                 aux→info = ∅; aux→next = *lista; *lista = aux;
                                 prec = *lista →next; corr = prec→next;
                               }
        else { prec = *lista;
               corr = prec → next; } .
```

- while (. . . .) pagina successiva

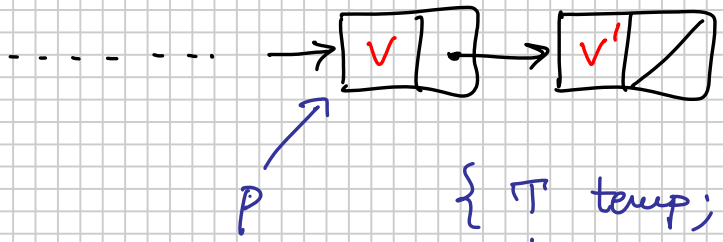
```
while ( corr != NULL )
  { if (corr -> info < ϕ)
      { ListaDiElementi aux = malloc (-----);
        aux -> info = ϕ ;
        aux -> next = corr;
        prec -> next = aux;
        prec = corr;
        corr = corr -> next;
      }
    else
      {prec = corr;
       corr = corr -> next;
      }
  }
}
prec = corr;
corr = corr -> next;
```



```
struct el { T info; struct el *next; }
typedef struct el ElementoDiLista;
typedef ElementoDiLista *listaDiElementi;
```

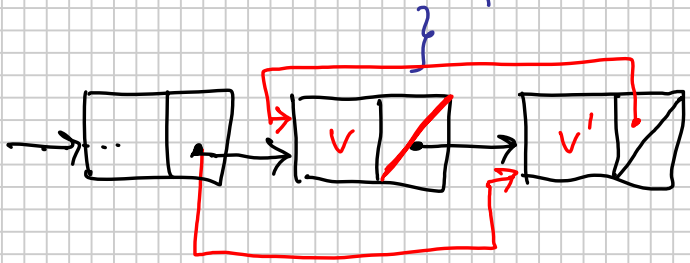
T è un tipo "generico"
in particolare non ho la certezza che su variabili di tipo T sia definito l'assegnamento.

Scrivere una procedura che cambia l'ordine degli ultimi 2 elementi di una lista data




```
{ T temp;
  temp = p->info;          (v)
  p->info = p->next->info; (v')
  p->next->info = temp;
}
```

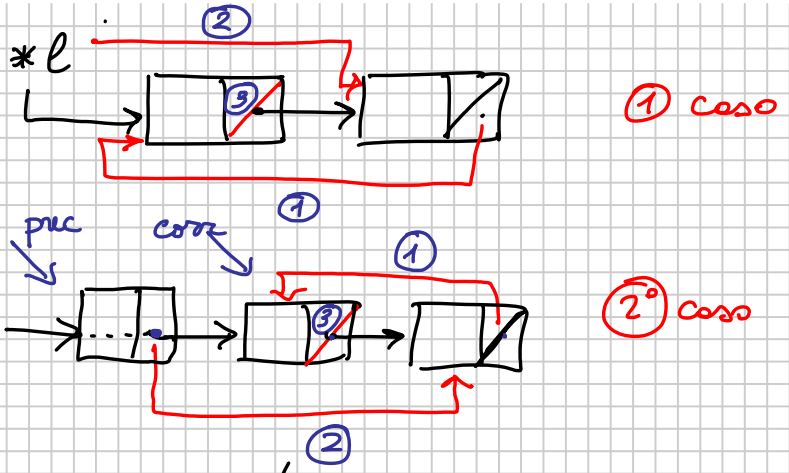
stiamo utilizzando l'assegnamento su variabili di tipo T



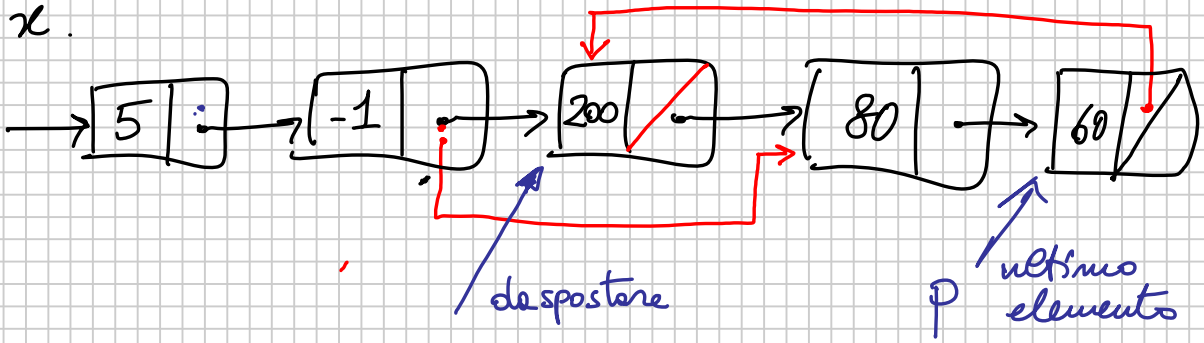
```
void scambiaup (ListaDiElementi *l)
{ if (*l != NULL)
    if (*l -> next != NULL)
      if (*l -> next -> next == NULL)
        { *l -> next -> next = *l;
          *l = *l -> next;
          *l -> next -> next = NULL; }
      else { ListaDiElementi prec = *l , corr = prec -> next;

        while (corr -> next -> next != NULL)
              { prec = corr; corr = prec -> next; }

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



Spostare in ultima posizione il primo elemento di una lista di interi maggiore di un valore dato x.

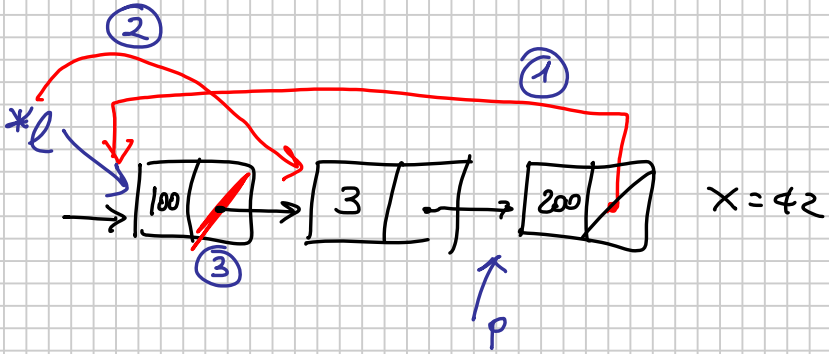


$x = 42$

NON USARE MALLOC.

Non serve creare nuovi elementi.

```
void sposta (ListaDiElementi *l, int x)
  { if (*l != NULL)
      if (*l -> next != NULL)
        { if (*l -> info) > x
            { ListaDiElementi p = *l;
              while (p->next!=NULL)
                { p = p -> next; }
              p -> next = *l;
              *l = *l -> next;
              p -> next -> next = NULL; }
```



$x = 42$

```
else /* il primo elemento della lista è <= x */
  { listaDiElementi toMove, prec, corr;
    toMove = NULL;  prec = *l;  corr = prec -> next;  int primo = 1;

    while (corr != NULL)
       { if (corr -> info > x && primo)
            { toMove = prec;  primo = 0; }

         prec = corr;
         corr = corr -> next;
       }
    if ( !primo )
       { prec -> next = toMove -> next;
         toMove -> next = toMove -> next -> next;
         prec -> next -> next = NULL;
       }
  }...
```