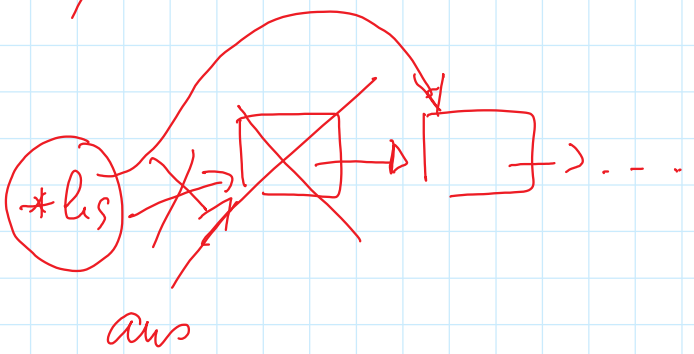
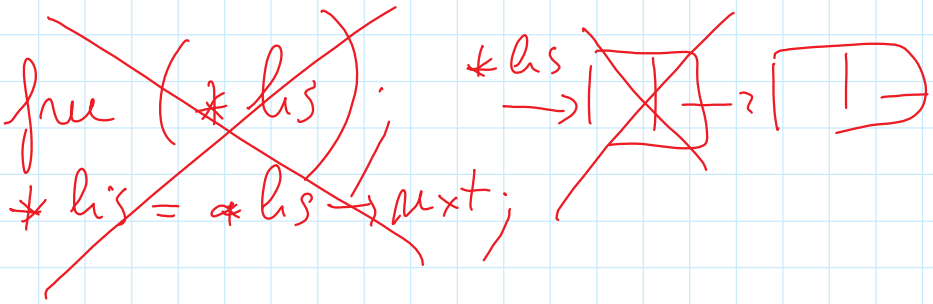
```
struct el
{ int info;
  struct el * next;
}

typedef struct el ElementoDiLista;
typedef ElementoDiLista * ListaDiElementi;
```

Procedura che cancella il primo elemento di una lista (se c'è).

```
void canc ( ListaDiElementi * lis)
{ if (*lis != NULL)
  { ListaDiElementi aus = *lis;
    *lis = *lis -> next;
    free (aus);
  }
}
```

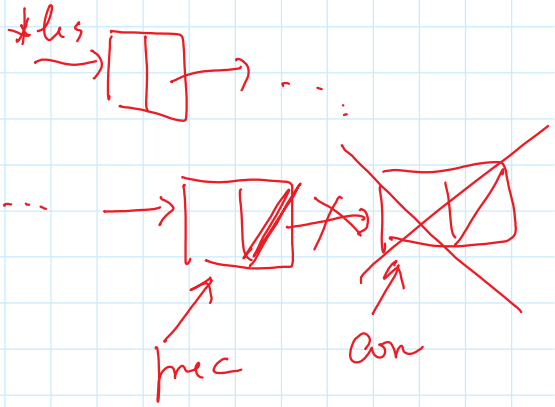




giovedì 9 novembre 2017 16:16

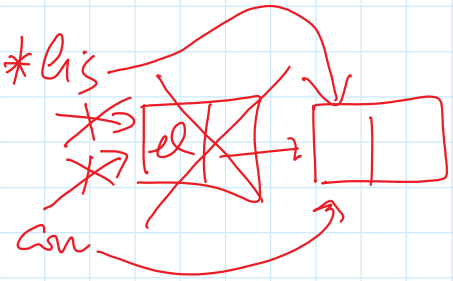
Cancellare l'ultimo elemento di una lista (se c'è)

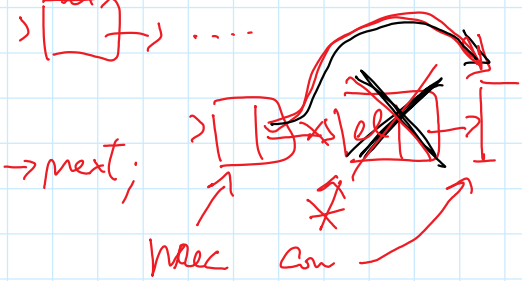
```
void canc (ListeDiElementi * lis)
{  if (*lst != NULL)
     { ListaDiElementi prec = NULL;
       ListeDiElementi corr = *lis;
       while (corr -> next != NULL)
          {  prec = corr;
             corr = corr -> next;
          }
       if (prec == NULL)
          { free (*lis);
            *lis = NULL;
          }
       else
          { free (corr);
            prec -> next = NULL;
          }
     }
}
```



giovedì 9 novembre 2017 16:26

```
void cancella (listaDiElementi *lis, int el)
{ if (*lis != NULL)
  { listaDiElementi *prec = NULL;
    listaDiElementi *corr = *lis;
    while (corr != NULL)
      if (corr->info == el)
      { if (prec == NULL)
        { *lis = *lis->next;
          free (corr);
          corr = *lis;
        }
        else
        { prec->next = corr->next;
          free (corr);
          corr = prec->next;
        }
      }
      else
      { prec = corr;
```





```
        cur = cur->next;
    }
}
```

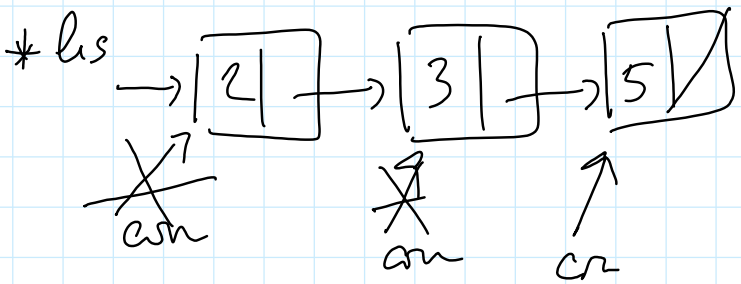
giovedì 9 novembre 2017 16:40

Funzione che restituisca la somma di tutti gli elementi di una liste di interi

```
int somma (ListaDiElement lis)
{ int s = 0;
  ListaDiElement corr = lis;
  while (corr != NULL)
  { s = s + corr -> info;
    corr = corr -> next;
  }
  return s;
}
```

corr -> next

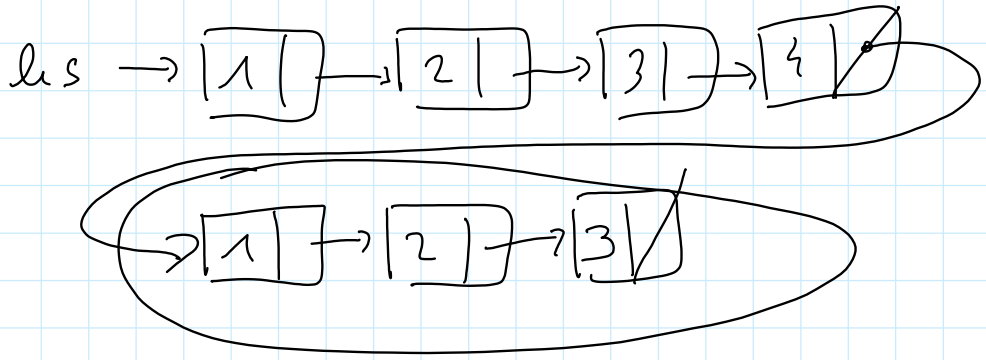
s = 2+3



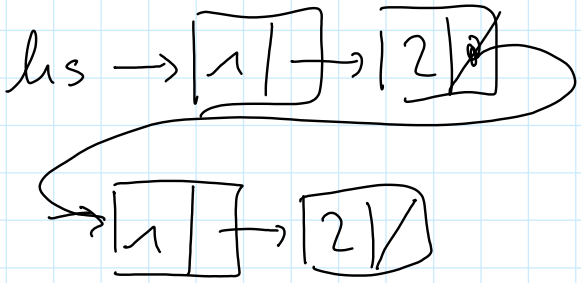
giovedì 9 novembre 2017 16:46

Si scriva una procedura che, presi una lista e un naturale n, crea una copia dei primi n elementi della lista e aggiunge tali elementi in coda alla lista stessa. Se la lista contiene meno di n elementi, tutti gli elementi vengono copiati e aggiunti in coda.

$n = 3$



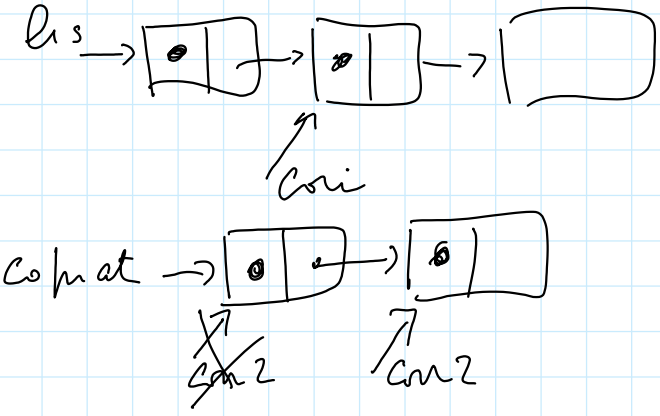
$n = 3$



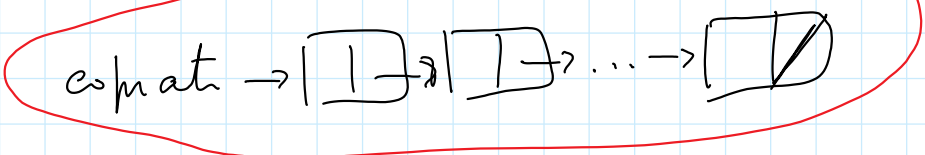
giovedì 9 novembre 2017 16:51

```
void aodd (ListaDiElementi lis, int n)
{ ListaDiElementi corr = lis;
  ListaDiElementi copiati = NULL;
  ListaDiElementi corr2 = NULL;
  copiati = malloc(...);
  copiati -> info = lis -> info;
  corr2 = copiati;
  corr = corr -> next;
```

{ if(lis != NULL && n > 0)



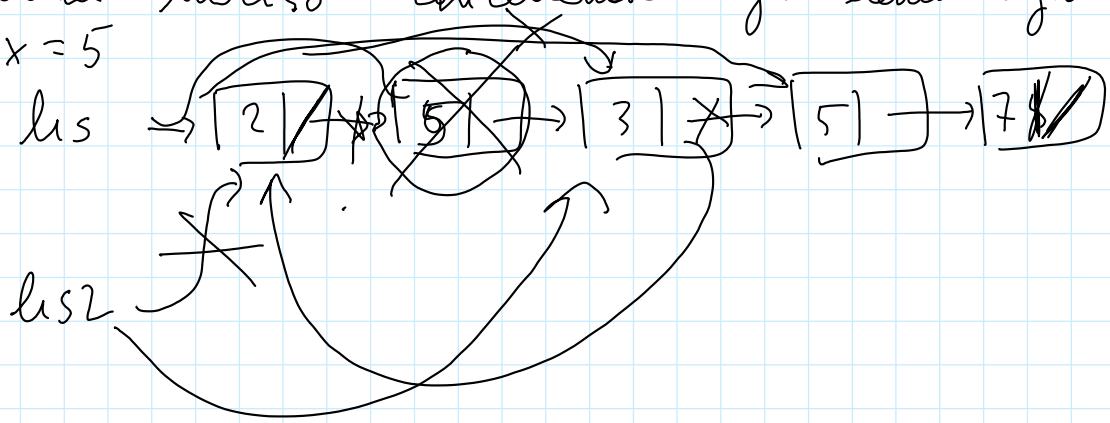

```
while( corr != NULL && n-1 > 0)
  { corr2 -> next = malloc(...);
    corr2 = corr2 -> next;
    corr2 -> info = corr -> info;
    corr = corr -> next;
    n = n - 1;
  }
corr2 -> next = NULL;
if (corr == NULL)
  { corr = lis;
```



```
    {  aux = lis;
       while (aux->next != NULL)
            aux = aux->next;
    }
else{  while (aux->next != NULL) ||
            aux = aux->next;
    }
  aux->next = copiat;
 }
}
```

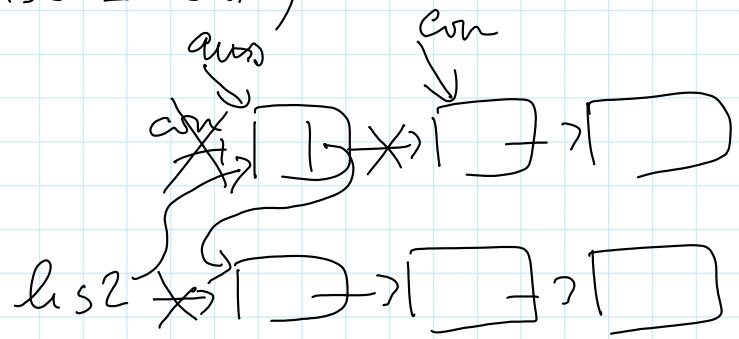
giovedì 9 novembre 2017 17:05

Scrivere una procedura che presa una lista e un intero x, ridispone gli elementi della lista in ordine inverso cancellando gli elementi uguali a x.



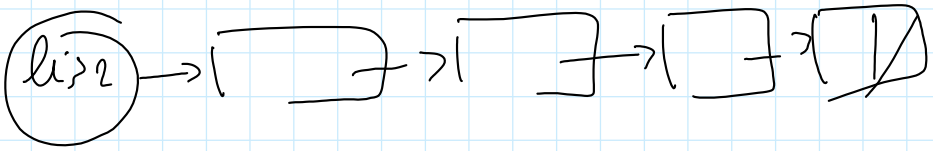

```
void move(ListeDiElement * lis, int x)
{ if (*lis != NULL)
    { ListeDiElement lis2 = NULL;
      ListeDiElement corr = *lis;
      while (corr != NULL)
        if (corr -> info == x)
          { ListeDiElement aus = corr;
            corr = corr -> next;
            free(aus);
          }
        else
          { ListeDiElement aus = corr;
            corr = corr -> next;
```

```
aux->next = lis2;
lis2 = aux;
}
```



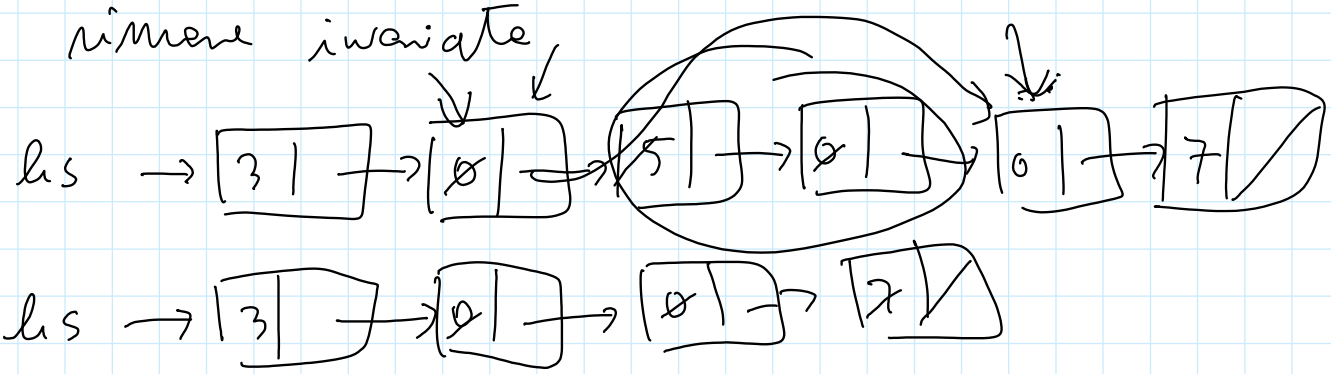

```
*lis = lis2;
}
}
```

curr = NULL



giovedì 9 novembre 2017 17:18

Scrivere una procedura C che, presa una lista elimina tutti gli elementi compresi tra il primo elemento $= \emptyset$ e l'ultimo elemento $= \emptyset$. Se ci sono meno di 2 elementi $= \emptyset$ la lista rimane invariata



giovedì 9 novembre 2017 17:23

```
void zeri (ListeDiElement lis)
{ ListeDiElement primo = NULL;
                 secondo = NULL;
//               corr = lis;
int trovato1 = 0;
int trovato2 = 0;
while (corr != NULL && !trovato1)
  if (corr->info == 0) trovato1 = 1;
  else corr = corr->next;
if (trovato1)
  { primo = corr;
    corr = corr->next;
    while (corr != NULL)
    { if (corr->info == 0)
        { trovato2 = 1;
          secondo = corr; }
      corr = corr->next;
    }
  if (trovato2)
    { while (primo->next != secondo)
```

```
if (lis != NULL)
  if (lis->next != NULL)
    if (lis->next->next
        != NULL)
```

```
{ lista_t_Element *aux = primo->next;
  primo->next = aux->next;
  free(aux);
}
}
}
...
}
```