martedì 13 dicembre 2016 09:17

ESERCITAZIONE PER LA PROVA IN ITINERE (E LO SCRITTO)

① → ESERCIZI SU LISTE IN C

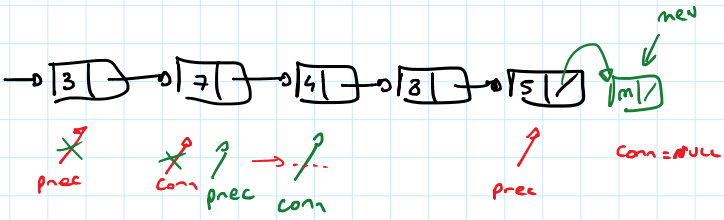
② → FUNZIONI RICORSIVE SU LISTE IN CAML

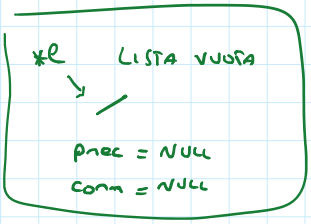
③ → FUNZIONI SU LISTE IN CAML SENZA RICORSIONE ESPLICITA

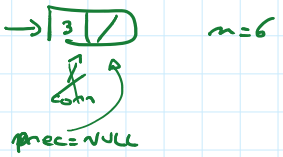
VEDERE TESTI (E SOLUZIONI) ANNI PRECEDENTI NELLA PAGINA WEB DEL CORSO

ESERCIZI SU LISTE IN C

- ES. 5 DEL 05/09/2014

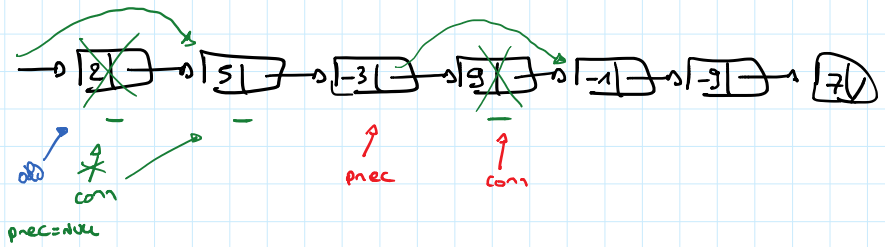







```
void inserisci (ListaDiElementi *l, int n)
{
    ListaDiElementi prec = NULL;
    ListaDiElementi conn = *l;
    int Trovato = 0;

    while (conn != NULL && !Trovato)
    {
        if (conn->info == n) Trovato = 1;
        else {
            prec = conn;
            conn = conn->next;
        }
    }
    if (!Trovato) {
        ListaDiElementi new = malloc (sizeof(ElementoDiLista));
        new->info = n;
        new->next = NULL;
        if (*l == NULL)          // LISTA VUOTA
            *l = new;
        else                     // LISTA NON VUOTA
            prec->next = new;
    }
}
```

ESERCIZIO 4 DEL 6/7/2016



n = 3

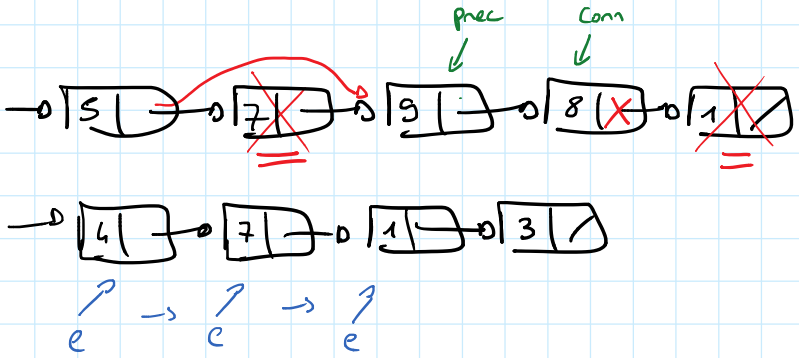
```
void cancella (ListaDiElementi *e, int n)
{
    ListaDiElementi prec = NULL;
    ListaDiElementi conn = *e;

    while (conn != NULL && n > 0)
    {
        if (conn -> info <= 0)
        {
            prec = conn;
            conn = conn -> next;
        }
        else {                                  // conn -> info > 0
            if (prec == NULL) {                 // conn == *e
                *e = conn -> next;
                free (conn);
                conn = *e;                      // non posso scrivere conn = conn->next
                                                // dopo la free!!
                n = n-1;
            }
            else {
                prec -> next = conn -> next;
                free (conn);
                conn = prec -> next;
                n = n-1;
            }
        }
    }
```

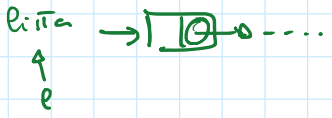
ALTERNATIVA

```
ListaDiElementi old = conn;
conn = conn -> next;
free (old)
```

3

ESERCIZIO 3 del 14/1/2016






```
int cerca (ListaDiElementi p, int n)
{   int Trovato = 0;
    while (p != NULL && !Trovato)
    {
        if (p->info == n) Trovato = 1;
        else p = p->next;
    }
    return Trovato;
}
```

|| FUNZIONE AUSILIARIA

```
void elimina (ListaDiElementi *p1, ListaDiElementi p2)
{
    ListaDiElementi prec = NULL;
    ListaDiElementi corr = *p1;

    while (corr != NULL)
    {
        if (cerca (p2, corr->info))
        {
            if (prec == NULL)          // primo elemento
            {
                *p1 = *p1->next;
                free (corr);
```

```
            conn = *p;
        }
        else {
            prec->next = conn->next;
            free (conn);
            conn = prec->next;
        }
    }
    else {
        prec = conn;
        conn = conn->next;
    }
  }
}
```

SUGGERIMENTI

1. pensate bene se la lista deve essere passata x valore o indirizzo
2. pensate bene a quanti/quali puntatori vi servono
3. fate attenzione ad assegnare i puntatori nell'ordine giusto
4. pensate bene ai casi particolari (lista vuota, lista con un solo elemento, ...)
5. ricordatevi di fare la free!!
6. fate delle simulazioni con carta e penna!

ESERCIZI SU FUNZIONI RICORSIVE SU LISTE (IN CAML)

- Data una lista di interi, costruire una coppia di liste in cui la prima lista contiene tutti gli elementi negativi e la seconda tutti i positivi (≥ 0)

split [5;7;-1;4;-3;-7;8] => ([-1;-3;-7],[5;7;4;8])

([...],[...])

L'ORDINE DI QUESTI ELEMENTI NON È IMPORTANTE

```
let rec split l =
    match l with
      [] -> ([],[])
    | x::xs -> let (l1,l2) = split xs in
                 if (x<0) then (x::l1, l2)
                          else (l1, x::l2) ;;
```

ESERCIZIO data una lista cancellare gli elementi ripetuti contigui

conc [8;5;5;7;4;4;4;5;9] => [8;5;7;4;5;9]

x y xs

```
let rec conc l =
    match l with
      []  -> []
    | x::[] -> x::[]
    | x::y::xs -> if x=y then conc y::xs
                  else x::(conc y::xs) ;;
```

x::[] ≡ [x]