Titolo nota 24/09/2015

Dato il tipo degli alberi binari `type 'a btree = Void | Node of 'a * 'a btree * 'a btree`

Si definisca una funzione

`canc : 'a btree -> ('a -> bool) -> 'a btree`

tale che `canc bt p` cancelli tutte le foglie il cui contenuto soddisfa `p`

Definire una funzione CAML `extract : 'a list list -> 'a list`

che, presa una lista di liste, `ll`, restituisca la lista contenente l'ultimo valore, se c'è, delle liste elementi di `ll`.

es: `extract [[3;4]; []; [1]; [2;3;6]] = [4;1;6]`

Titolo nota 24/09/2015

Indicare il tipo delle seguenti funzioni:

```
let f g h x = if g x then h x else 1;;
let f g h = g h + h 3;;
let f x = let g y = if y then x else x+1
          in g;;
```

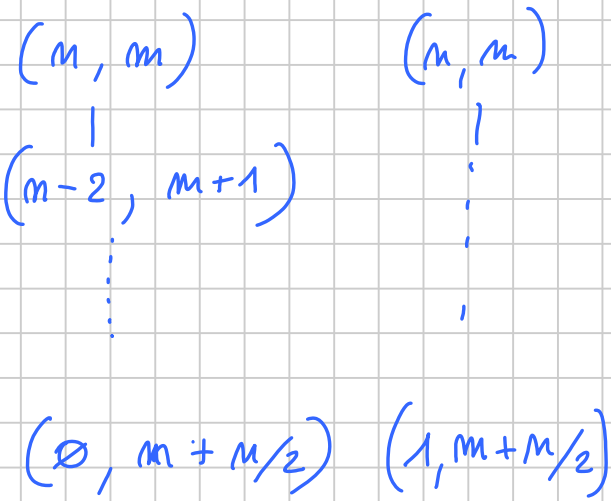
Scrivere una procedura C che cancella il primo e l'ultimo elemento di una lista, se esistono.

Titolo nota 24/09/2015

Definire una funzione ricorsiva $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ tale che $(\forall x, y \in \mathbb{N} \,.\, f(x,y) = 3x + y + 4)$ in modo che la relazione di precedenza usata nella definizione sia:

$$(\forall x, y, x', y' \in \mathbb{N} \,.\, (x,y) \sqsubset (x',y') \equiv x' = x + 2 \wedge y = y' + 1)$$

$$f(n,m) = \begin{cases} \vdots \\ f(n-2, m+1) \cdots \end{cases}$$



Titolo nota 24/09/2015

```
let rec cauc bt p = match bt with
        Void -> Void
      | Node (x, Void, Void) when p x -> Void
      | Node (x, Void, Void) when not (p x) -> Node (x, Void, Void)
      | Node (x, lbt, rbt) when lbt <> Void or rbt <> Void
                -> Node (x, cauc lbt p, cauc rbt p) ;;
```

Titolo nota 24/09/2015

```
let rec extract ll =
  let rec ultimo l = match l with
      [x] -> x
    | x::y::ys -> ultimo (y::ys)
  in match ll with
      [] -> []
    | []::yss -> extract yss
    | (x::xs)::yss -> ultimo (x::xs) :: extract yss ;;
```

Titolo nota 24/09/2015

```
let rec extract ll =
    let rec ultimo l = match l with
        [] -> []
      | [x] -> [x]
      | x::y::ys -> ultimo (y::ys)
    in match ll with
      [] -> []
    | xs::xss -> ultimo xs @ extract xss ;;
```

Titolo nota 24/09/2015

```
let extract ll =
    let rec ultimo l = match l with
        [] -> []
      | [x] -> [x]
      | x::y::ys -> ultimo (y::ys)
    in let f x y = ultimo x @ y

    in foldr f [] ll ;;
```

Titolo nota 24/09/2015

Indicare il tipo delle seguenti funzioni:

```
let f g h x = if g x then h x else 1;;
let f g h = g h + h 3;;
let f x = let g y = if y then x else x+1
          in g;;
```

f: ('a → bool) → ('a → int) → 'a → int

f: ((int → int) → int) → (int → int) → int

f: int → bool → int

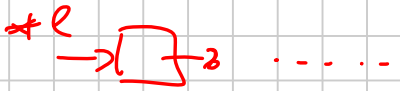
Titolo nota 24/09/2015

```
void conc (ListeDiElementi * l)
{ if (*l != NULL)
    if ((*l)->next == NULL) { ListeDiElementi corr = *l;
                              *l = (*l)->next;
                              free (corr);
                            }
    else if ((*l)->next->next == NULL) { free ((*l)->next);
                                         free (*l);
                                         *l = NULL
                                       }
    else
```

*l = NULL;

{ free (*l);
  *l = NULL;
}

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



Titolo nota 24/09/2015

$$f(n,m) = \begin{cases} m+4 & \text{se } n=0 \\ m+7 & \text{se } n=1 \\ f(n-2, m+1)+5 & \text{se } n>1 \end{cases}$$

Casi base: ovvi

Caso induttivo: $f(n-2, m+1) = 3(n-2) + m + 1 + 4 \Rightarrow f(n,m) = 3n + m + 4$

$f(n,m)$

$= \{$ def $f,\ n>1\}$

$f(n-2, m+1) + 5$

$= \{$ ip. induttiva $\}$

$3 \cdot (n-2) + m + 1 + 4 + 5$

$= \{$ calcolo $\}$

$3n - 1 + m + 5$

$= \{$ calcolo $\}$

$3n + m + 4$