SIMULAZIONE DI VERIFICA INTERMEDIA

- SOLUZIONI -

ES 1

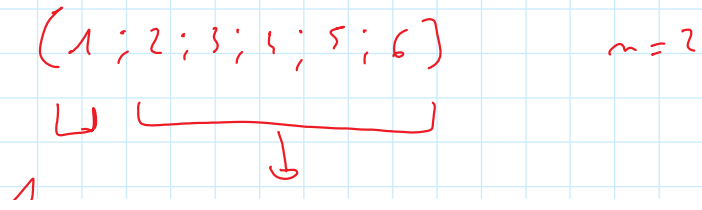
```
let rec cancella e =
    match e with
      [] -> []
    | x::[] -> []
    | x::y::[] -> []
    | x::xs -> x::(cancella xs) ;;
```

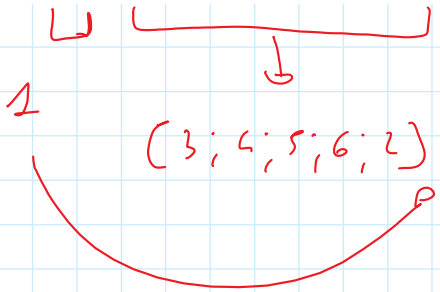
ES 2

```
let rec sposta n lis =
    match (n, lis) with
        (0, l) -> l
      | (n, []) with n>0 -> []
      | (n, x::xs) with n>0 ->
                (sposta n-1 xs) @ [x] ;;
```

ALTRA SOLUZIONE PIU' EFFICIENTE

```
let sposta n lis =
    let rec sposta_acc n l1 l2 =
        match (n, l1) with
            (0, l) -> l @ l2
          | (n, []) with n>0 -> l2
          | (n, x::xs) with n>0 ->
                sposta_acc n-1 xs x::l2
    in
        sposta_acc n lis [] ;;
```





m=1

ES 3

```
let Take n lis =
    let length l =
        let f x y = 1 + y
        in foldr f 0 l
    in
        let m = length lis - n
        in
            let g x (l1, k) =
                if k>0 then (l1, k-1)
                       else (x::l1, k)
            in
                let (l2, w) = foldr g ([], m) lis
                in l2 ;;
```

[1; 2; 3; 4; 5; 6] n = 2

[1; 2]

ALTRA SOLUZIONE (più COMPLICATA)

```
let Take n lis =
    let cancellaultimo l =
        let f x (y1,y2) =
            if y2 then (x::y1, y2)
                  else (y1, True)
        in
        let (l1,b) = foldr f ([],false) l
        in l1
    in
    let g x (l1,m) =
        if m<n then (x::l1, m+1)
               else
        (x::(cancellaultimo l1), m)
    in
    let (l2,k) = foldr g ([],0) lis
    in l2 ;;
```

ES 4

```
let cancellacoppie l =
    let p (x1,x2) = x1 + x2 <> 10
    in
        filter p l ;;
```