```
Esercizio 1 - Soluzione
(a)
module type TAB =
 sig type 'a tab
  val mk: int * int -> 'a tab
  exception OutOfBoundsException of string
  exception IllegalArgumentException
  val set: 'a tab -> (int * int) -> 'a option -> 'a tab
  val get: 'a tab -> (int * int) -> 'a option
  val size: 'a tab -> (int * int)
end;;
(b)
module Tab =
(struct
   open List
   type 'a tab = 'a option list list
 (* Presentazione Valore: Un valore 'a tab e' [r1;...;rn] dove ri = [vi1;...;vim]
   sono le n≥0 righe della tabella, ognuna avente m≥0 valori 'a option.
   AF(c) = [r1;...;rn] dove vij = nth (nth c i) j (forall i\in[1..n], j\in[1..m])
   I(c) = (c = [r1;...;rn] && n > 1) => length ri = length rj (forall i,j\in[1..n])
   con: [,], &&, length delimitatori di liste e operazioni OCaml
 *)
 (* definizione eccezioni pubbliche *)
 (* definizione delle operazioni ausiliarie e pubbliche *)
 end:TAB);;
(c)
   exception OutOfBoundsException of string
   exception IllegalArgumentException
   let mk(n,m) = let rec nOfv k v = if k=0 then [] else v::(nOfv(k-1)v)
                 in if (n<0 || m<0) then  raise(IllegalArgumentException)
                    else nOfv n (nOfv m None)
(d)
   let inBounds tab (n,m) = match tab with
      [] -> n=0 && m=0
      |x::xs -> (n>0 && n<=1+length xs) && (m>0 && m<=length x)
   let set tab (n,m) v = if inBounds tab (n,m) then raise (OutOfBoundsException "set")
                         else let rec replace t k v = if k=0 then v::(tl t)
                                                      else (hd t)::(replace t(k-1)v)
                              in replace tab n (replace (nth tab n) m v)
   let get tab (n,m) = if not(inBounds tab (n,m)) then raise(OutOfBoundsException "set")
                       else nth (nth tab n) m


Esercizio 2 - Soluzione
(a)
    public Tab(int n, int m) throws IllegalArgumentException;
    public void set(int i, int j, A v) throws OutOfBoundsException;
    public A get(int i, int j) throws OutOfBoundsException;
    public IntPair size();
(b)
public class TabE<A> extends Tab<A>{
    private Vector<A> In;
    /*
     Presentazione Valore: la stessa della super classe Set<A>
     AF(c) = AF(c.super())
     I(c) = forall v\in [|A|]:
            c.In.contains(v) iff ((v!=null)&&(v.equals(get(i,j)) (for some i,j)))
    */
    /* definizione metodi ausiliari e pubblici */
}
(c)
```

```
    public TabE(int n, int m) throws IllegalArgumentException{
        super(n,m);
        In = new Vector<A>();
    }
    public void set(int i, int j, A v) throws OutOfBoundsException{
        if (v!=null && In.contains(v)) return;
        int N = size().getL(); int M = size().getL();
        if (i<1 || i>N || j<1 || j>M) throw new OutOfBoundsException("Tab:set");
        A temp = get(i,j);
        if (temp!=null) In.remove(temp);
        if (v!=null) In.add(temp);
        super.set(i,j,v);
    }
(e)
    public TabE<A> clone() throws CloneNotSupportedException{
        TabE<A> ret = (TabE<A>) super.clone();
        ret.In = (Vector<A>) In.clone();
        return ret;
    }
```