```
Esercizio 1 - Soluzione
(a)
module type QUEUE =
 sig type 'a queue
  val mk: 'a list -> 'a queue
  exception InvalidArgException of string
  val isEmpty: 'a queue -> bool
  val get: 'a queue -> 'a
  val add: 'a queue -> 'a -> 'a queue
  val remove: 'a queue -> 'a -> 'a queue
end;;

(b)
module Queue =
(struct
   type 'a queue = Q of 'a list * 'a list
 (* Presentazione Valore: Un valore 'a queue è <e1,...,en< dove ei sono gli n≥0 elementi
                          con e1 primo inserito e primo da estrarre (FIFO da sx a dx)
    AF(c) = << iff c == Q(in,out) && length(in @ out) = 0
    AF(c) = <x1,...,xn,y1,...ym< iff c == Q(in,out) && length(in @ out) > 0 &&
                                     out == [x1,...,xn] && (rev in == [y1,...,ym])
    I(c) = Sia c == Q(in,out), allora:  (out==[] => in==[]
    con: [,], &&, @, rev, delimitatori di liste e operazioni OCaml
 *)

 (* public and auxiliary definitions *)
end:QUEUE);;

(c)
   let mk xx = Q([],xx)
   exception InvalidArgException of string

(d)
   let remove (Q(in,out)) z = match(in,out) with
            |(_,x::xs) when (x = z) -> if (xs <> []) then Q(in,xs) else Q([],List.rev xs)
            |(_,x::xs) when (isIn z xs) -> Q(in,x::rem z xs)
            |(yy,x::xs) when (isIn z yy) -> Q(List.rev(rem x (List.rev yy)),out)
            |_ -> (Q(in,out))



Esercizio 2 - Soluzione
(a)
    public QueueE(int k)throws IllegalArgumentException;
```

```
45              // inizializza solo a code vuote per max. k≥0 elementi diversi
46      public Boolean isIn(A e);
47      public int waitUp(A e)throws IllegalArgumentException;
48      public void removeAll(Vector<A> out);
49
50  (b)
51  public class QueueE<A> extends Queue<A> implßements Cloneable{
52      private Vector<A> els;
53      private int maxSize;
54       /*
55       Presentazione Valore: come nella classe Queue<A> data.
56       AF(c) = AF(c.super())
57       I(c) = (0≤c.els.size()≤c.maxSize) &&
58              (PerOgni oggetto o) (c.els.remove(o) ? !(c.els.contains(o)) : true)
…  == true,
59       nota: null può essere incluso in c.els come ogni altro valore di tipo A
60       */
61    //public and auxiliary definitions
62  }
63
64  (c)
65      //solo add e remove richiedono overriding.
66      public void add(A e){
67          if (isIn(e) || (els.size() == maxSize)) return;
68          super.add(e); els.add(e);
69      }
70      public void remove(A e){
71          super.remove(e); els.remove(e);
72      }
73
74  (d)
75      public QueueE(int k) throws IllegalArgumentException{
76          super(new Vector<A>());
77          els = new Vector<A>();
78          if (k≥0) {
79              maxsize = k; return;
80          }
81          maxsize = 0; // creata una coda con maxsize pari a 0
82          throw new IllegalArgumentException();
83                      // notifica della anomalia
84      }
85
86      public Boolean IsIn(A e){
87          return(els.contains(e));
88      }
89      public int waitUp(A e) throws IllegalArgumentException {
90          for(int i=0; i<els.size(); i++){
91              if(els.get(i).equals(e)) return(i);
92          }
```

```
93          throw new IllegalArgumentException();
94      }
95      public void removeAll(Vector<A> ee) {
96          if(ee == null || ee.size() == 0) return;
97          for(int i=0; i<ee.size(); i++){
98              A u = ee.get(i);
99              remove(u);
100         }
101     }
102
103 // IllegalArgumentException è predefinita ed è qui, usata per brevità
104
105 (e)
106     //solo clone richiede overriding
107     public QueueE<A> clone() throws CloneNotSupportedException{
108         QueueE<A> ret = (QueueE<A>) super.clone();
109         ret.els = new Vector<A>();
110         for(int i=0; i<els.size(); i++) {
111             ret.els.add(els.get(i));
112         }
113         return ret;
114     }
115
116
117
```