QUESTION 1

(a) The abstract data type of the "handshake set" is represented using an array "value" in which the first "size" elements are the numbers from the set, which must be distinct and in the interval [0..MAX) and by an array "loc", which points to the places where each value is in value, or they have (-1) as a sign that the value is not in the set.

```
trait IntSet {

    // State: set ∈ P([0..MAX)) - the powerset of [0..MAX)
    // Init: set = {}

    /* test if x is a member of the set */
    // Pre: 0 ≤ x < MAX
    // Post: return (loc(x) != -1) && (size > loc(x)) && (value(loc(x)) == x), since the location should indicate a
position within [0..size) to mean that x appears in value and on that position we should find x
    def contains (x: Int): Boolean

    /* insert x into the set */
    // Pre: 0 ≤ x < MAX && size < MAX
    // Post: size = size_0 + 1 && loc(x) = size - 1 && value(size - 1) = x (only if x was not in the set before)
    def insert (x: Int)

    /* delete x from the set */
    // Pre: 0 ≤ x < MAX
    // Post: size = size_0 - 1 && loc(x) = -1 && value(loc(x)_0) = value(size) &&
            loc(value(size)) = loc(x)_0 → we replace the value we delete with the last element
            from value (happens only if x was in the set before)
    def delete (x : Int)

    /* make the set empty */
    // Post: size = 0
    def clean ()
}
```

The client who uses this class must obey the preconditions that are imposed, mainly to not test, add or delete any value that is greater or equal than MAX and the implementer should warn him/her if they do that and also to not add too many elements.

(b)

Abstraction function:

$$set = \{ value(i) \mid loc(value(i)) = i, value(i) \in [0..MAX), i \in [0..size) \}$$

ΔTi: $0 \leq value[0..size) < MAX \;\&\&\; 0 \leq size \leq MAX \;\&\&\; -1 \leq loc(i) < MAX$

The ΔTi imposes constraints on the arrays we are working with and the abstraction functia shows that the abstract object, the set, has the necessary properties given by the initial conditions of the problem.

(c)

```
class Handshake extends intSet {
    private var MAX = 1000
    private var loc = new Array[Int](MAX)
    for (i <- 0 until MAX) loc(i) = -1
    private var value = new Array[Int](MAX)
    var size = 0
```

(ii) // an integer x is in set if its location exists and it is pointing to an index that is less than the current number of elements in the set and if we have value(loc(x)) == x

```
    def contains(x : Int) : Boolean = (loc(x) != -1) && (size > loc(x)) && (value(loc(x)) == x)
```

// When the result is false, we either have not even inserted the element or have deleted it before, in which case loc(x) = -1, or if we cleared the array before and the location where x is pointed to be at is bigger than the number of elements we have, in which case size ≤ loc(x), or when we added a lot of elements after the clear, but none of them is x and, although loc(x) < size, we do not have value(loc(x)) == x.

```
    def insert(x : Int) = if (contains(x) == false) {
        loc(x) = size
        value(size) = x
        size += 1
    }
```

(iii) // if the element we want to delete is the last one, meaning loc(x) = size - 1, we simply swap it with itself and delete it; in general we swap the element we want to delete with the last one, so that only the first size elements of value are actually in set and no others

```
    def delete(x: Int) = if (contains(x)) {
        value(loc(x)) = value(size - 1)
        loc(value(size - 1)) = loc(x)
        loc(x) = -1
        size -= 1 }
```

(i) // the definition of clean is correct, since if size becomes 0, we have that all the contains(x) are false from the start, $0 \leq x < MAX$

```
    def clean () = size = 0
}
```