```
/*
Versione 1 - vedi SA.
Una versione C del QSort definito utilizzando la Metodologia della Programmazione Strutturata
Vedere il relativo materiale discusso nelle lezioni di ... 2015 a Matematica.
Vedere il progetto e le motivazioni di questa versione C, prima di studiarla.
*/

#include <stdio.h>
#include <stdlib.h>
#include "seq.h"


int ord(int t1, int t2){
        //un parametro di QSort, e di altre operazioni coinvolte in QSort.
        /* un'ordinamento sugli interi scelto tra gli infiniti possibili */
    return (t1<t2);
}

int eqOrd(int x, int y){
        //un parametro di QSort, e di altre operazioni coinvolte in QSort.
        /* operazione di equivalenza su valori semi-ordinati */
    return ((x==y)||(ord(x,y)&&ord(y,x)));
}

int separatore(intSeq u){//privata di QSort
        // dovrebbe avere anche il parametro per l'odinamento
        /* il più semplice: il primo minore se ripetizioni.
           restiruisce il minore dei primi due ordinabili */
    if (size(u)!=0){
        int x = tail(u);
        u = head(u);
        while (u!=NULL) {
            int y=tail(u);
            if (ord(x,y)&!eqOrd(x,y)) return y;
            if (ord(y,x)&!eqOrd(x,y)) return x;
            u = head(u);
        }
    }
    return 0; /* avrebbe dovuto sollevare eccezione (certo non rimanere
                (indefinita, perchè il separatore non serve in questo caso) */
}

intSeq divisioneLE(intSeq u, int a){//privata di QSort
    // dovrebbe avere anche il parametro per l'odinamento
    /* implementata come una funzione filtro: filtra i minori o equivalenti ad a */
    if (size(u)==0) return u;
    else {
        int x = tail(u);
        if (eqOrd(x,a)||ord(x,a)) {
//      if ((x==a)||ord(x,a)) {
            return add(divisioneLE(head(u),a),x);
        }
        else return divisioneLE(head(u),a);
    }
}

intSeq divisioneGT(intSeq u, int a){//privata di QSort
    // dovrebbe avere anche il parametro per l'odinamento
    /* implementata come una funzione filtro: filtra i maggiori di a */
    if (size(u)==0) return u;
    else {
```

```
        int x = tail(u);
        intSeq r = divisioneGT(head(u),a);
        if (ord(a,x)&!eqOrd(x,a)) return add(r,x);
        return r;
    }
}

intSeq QSort(intSeq cc){
    /* dovrebbe avere il parametro per l'ordinamento. In C non possaimo trasmettere una
       funzione, quindi dobbiamo accedere l'ordinamento con un "trucco": Lo trattiamo come
       un valore funzione esterno. Ovviamente se vogliamo usare un differente ordinamento
       dobbiamo cambiare tutte le definizioni: sepatore, divisioneLE, divioneGT, QSort.
    */
    /* Qui dovrebbero essere poste le funzioni private:
     int separatore(...){...};
     Seq(int) divisioneLE(...){...};
     Seq(int) divisioneGT(...){...};
     in C le procedure/funzioni non sono Fully Abstract.
    */
    if (size(cc)<2) return cc;
             /* size fa parte della definizione di Seq(int) */
    {
        int a = separatore(cc);
                /* Pasticcio: abbiamo nascosto il parametro ord */
        intSeq qle = divisioneLE(cc,a);
                /* Pasticcio: abbiamo nascosto il parametro ord */
        intSeq qgt = divisioneGT(cc,a);
                /* Pasticcio: abbiamo nascosto il parametro ord.
                   Cambiare ord impone la revisione e riscrittura di tutte
                   le funzioni significative: QSort, separatore, divisione. */
        if (size(qle)==0) return qgt;
        if (size(qgt)==0) return qle;
        return append(QSort(qle),QSort(qgt));
    }
}
```