```
/*
SottoSequenza (ad allocazione dinamica) [di interi].
SottoSequenza NON MODIFICABILE.
*/

#ifndef MUTSUBS
#define MUTSUBS

#include <stdio.h>
#include <stdlib.h>
#include "bool.h"
#include "mutS.h"

struct CoppiaIntMutS {//sottosequenze: richiede prop. di integrità
    intMutS inf;
    intMutS sup;
};

typedef struct CoppiaIntMutS iPair;
typedef iPair * intMutSubS;

/* ------------ OPERAZIONI ------------------------------ */


/* ------------ OSSERVAZIONE COMPONENTI ----------------- */

bool innerIntegrityCheck(intMutS inf, intMutS sup){//privata
    // privata di integrityCheck
    /* controlla che inf preceda sup */
    if (inf == sup) return true;
    if (inf == NULL)return false;
    return innerIntegrityCheck(inf->next,sup);
};

bool integrityCheck(intMutS inf, intMutS sup){//privata
    // privata di mkSubS
    /* controlla che inf preceda sup */
    if (inf == sup) return true;
    if ((inf == NULL)||(sup == NULL)) return false;
    return innerIntegrityCheck(inf,sup);
};

/* ------------ COSTRUZIONE DELLA STRUTTURA -------------- */

intMutSubS mkSubS(intMutS inf, intMutS sup){
        /* crea un sottosequenza */
        /* integrità: Richiede che inf preceda sup */
    intMutSubS subS;
    if (!integrityCheck(inf,sup)) return NULL; //integrità
    subS = (intMutSubS)malloc(sizeof(iPair));
    subS->inf = inf;
    subS->sup = sup;
    return subS;
};

/* ------------ OSSERVAZIONE COMPONENTI ----------------- */

intMutS inf(intMutSubS u){//pubblica
    /* restituisce estremo inferiore sottosequenza u */
    if (u!=NULL) return u->inf;
```

```
    return 0;
};

intMutS sup(intMutSubS u){//pubblica
    /* restituisce elemento superiore sottosequenza u */
    if (u!=NULL) return u->sup;
    return 0;
};

/* ---------------------- I/O ----------------------------- */

intMutSubS readIntSubS(){//pubblica
        /* ausiliaria per I/O */
        /* sequenza di coppie intero ',' con delimitatore sinistro '[',
           e delimitatore destro, un qualunque carattere bv */
    intMutS inf = empty();
    intMutS sup = empty();
    char c;
    if ((scanf("%c",&c)!=1)||(c!='[')){
        printf("Carattere inatteso: la sequenza deve avere la forma: [xx,...,xxc\n");
        return mkSubS(inf,sup);
        }
    {/* primo elemento */
        int n;
        int endTest = 1; // 1 per true
        while (endTest && (scanf("%d",&n)==1)){
            sup = addTail(sup,n);
            if (isEmpty(inf)) inf = sup;
            endTest = scanf("%c",&c)==1 && (c==',');
        }
        if (endTest) scanf("%c",&c);
        return mkSubS(inf,sup);
    }
}

void innerWriteDelimited(intMutS inf, intMutS sup){//privata
    /* locale e privata di writeIntSeq */
    if (inf == sup) return;
    printf("%d,",val(inf));
    innerWriteDelimited(moveR(inf),sup);
    return;
};

void writeDelimitedIntSeq(intMutS inf, intMutS sup){//pubblica
    /* ausiliaria per I/O */
    /* qui definizione di innerWrite */
    if (isEmpty(sup)){
        printf("[]\n");
        return;
    }
    printf("[");
    innerWriteDelimited(inf,sup);
    printf("%d]\n",val(sup));
    return;
};

void writeIntSubS(intMutSubS sub){//pubblica
    if (sub == NULL){
        printf("[]\n");
        return;
```

```
    }
    writeDelimitedIntSeq(sub->inf, sub->sup);
    return;
}

#endif
```