Struttura dati misteriosa

Il programma in calce contiene l'implementazione di una struttura dati classica, qui chiamata genericamente `dataStructure`. Il programma contiene anche una funzione `main` che può essere usata per fare dei test assieme ai file di input e output messi a disposizione.

Analizzate il programma e rispondete alle domande seguenti, inserendo le vostre risposte in commenti all'interno del programma.

1. Individuate di che struttura dati si tratta.
2. Descrivete come è implementata tale struttura dati.
3. Che tipo di struttura dati è `struct x` e cosa rappresenta in particolare in relazione a `dataStructure`? Scegliete dei nomi più significativi per i tipi `struct x` e `dataStructure`
4. Cosa rappresenta il membro `A` di `dataStructure`?
5. Cosa fanno le funzioni `dataStructure_insert` e `add`?

Programma da analizzare e commentare

```
/* Di che struttura dati si tratta e come è implementata in questo progra
INSERITE QUI LA VOSTRA risposta

...

*/



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


struct x {
  struct x *next;
  int v;
};

/* Come descrivereste struct x?
INSERITE QUI LA VOSTRA risposta
```

```
...

*/
struct dataStructure {
  int n, m;
  struct x **A;
};

/* Cosa rappresenta il membro A di dataStructure?
INSERITE QUI LA VOSTRA risposta

...

*/


typedef struct dataStructure *Ds;



/* creazione della struttura dati */
Ds dataStructure_new( int n );

/* inserimento di ... nella struttura dati */
/* Documentate QUI fa la funzione dataStructure_insert
... */
void dataStructure_insert( Ds s, int v, int w );

/* lettura da standard input */
Ds dataStructure_read( void );

/* stampa su standard output */
void dataStructure_print( Ds s );









// FUNZIONI AUSILIARIE
//*************************************

/* DOCUMENTATE QUI cosa fa la funzione add
... */
struct x *add( struct x *l, int v ) {
```

```c
    struct x *new = malloc( sizeof( struct x ) );
    new -> v = v;
    new -> next = l;
    return new;

}



// MAIN
// ***********

int main(){
    Ds s = NULL;
    s = dataStructure_read();

    dataStructure_print(s);

    return 0;
}


// IMPLEMENTAZIONE DELLE FUNZIONI DELL'INTERFACCIA
// ***********************************************************************

Ds dataStructure_new( int n ){

    Ds s = malloc(sizeof(struct dataStructure));
    if(!s) {
            fprintf(stderr,"Errore di Allocazione\n");
            exit(EXIT_FAILURE);
    }
    s-> n = n;
    s -> A = calloc( n, sizeof( struct x* ) ); //
    return s;
}


void dataStructure_insert( Ds s, int v, int w ){
        s->A[v] = add(s->A[v],w); //
        if ( v != w )
        s->A[w] = add(s->A[w],v);
}

Ds dataStructure_read( void ){
    int n, m, i, v, w;
    Ds s;
```

```
    scanf( "%d", &n );
    s = dataStructure_new( n );

    scanf( "%d", &m );
    s -> m = m;
    for (i = 0; i < m; i++) {
        scanf( "%d %d", &v, &w );
        dataStructure_insert( s, v, w);
    }

    return s;
}

void dataStructure_print( Ds s ){
    printf( "\n************** stampa\n" );
    int n = s -> n;
    struct x **al = s -> A;
    struct x *curr;

    printf( "n = %d\nm = %d\n", n, s -> m );

    for ( int i = 0; i < n; i++ ) {
        printf( "%d:", i );
        for ( curr = al[i]; curr != NULL; curr = curr -> next ) {
            printf( " %d", curr -> v );
        }
        printf( "\n" );
    }
    printf( "fine stampa  ************************\n\n");

}
```