```
/* Completare l'esercizio implementando le funzioni mancanti in stile ricorsivo per il
calcolo della lunghezza di un alista concatenata e di uguaglianza tra liste.

Se ad esempio l'input del programma è il seguente:

lista 1:

3

5

7

9

lista 2:

5

7

9

0

L'output del programma sarà:

Ecco la lista 1: [3 5 7 9 ]
Ecco la lista 2: [5 7 9 0 ]
La lunghezza della lista 1 è : 4
La lunghezza della lista 2 è : 4
Le due liste NON sono uguali!


 */



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

struct entry {
    int value;
    struct entry * next;
};

struct entry *construct_list();
void list_print(struct entry *this);
int list_length(struct entry *this);
bool list_equal(struct entry *this, struct entry *that);


int main(void) {
    printf("Inserisci i valori della prima lista: ");
    struct entry * l1 = construct_list();

    printf("Inserisci i valori della seconda lista: ");
    struct entry * l2 = construct_list();

    printf("Ecco la lista 1: ");
    list_print(l1);

    printf("Ecco la lista 2: ");
    list_print(l2);
```

```c
    printf("La lunghezza della lista 1 è : %i\n", list_length(l1));
    printf("La lunghezza della lista 2 è : %i\n", list_length(l2));

    printf("Le due liste %s sono uguali!\n", (list_equal(l1, l2) ? "" : "NON")  );
    return 0;
}


//Dichiarazione e definizione funzioni:
struct entry * construct_list(){
    int n1, n2, n3, n4;
    scanf("%i", &n1);
    scanf("%i", &n2);
    scanf("%i", &n3);
    scanf("%i", &n4);

    struct entry *pn1 = (struct entry*)malloc(sizeof(struct entry));
    struct entry *pn2 = (struct entry*)malloc(sizeof(struct entry));
    struct entry *pn3 = (struct entry*)malloc(sizeof(struct entry));
    struct entry *pn4 = (struct entry*)malloc(sizeof(struct entry));

    pn1->value = n1;
    pn2->value = n2;
    pn3->value = n3;
    pn4->value = n4;

    pn1->next = pn2;
    pn2->next = pn3;
    pn3->next = pn4;
    pn4->next = NULL;
    return pn1;

};

void list_print_aux(struct entry *this){

    if(this == NULL)
    {
        return;
    }
    else {
        printf("%i ", this->value);
        list_print_aux(this->next);
    };
};

void list_print(struct entry *this){ //supporta liste vuote
    printf("[");
    list_print_aux(this);
    printf("]\n");
};



int list_length(struct entry *this){ //supporta liste vuote
    /*
    implementare in stile ricorsivo
    */

    if( this == NULL ){

        return 0;
    }

    return 1 + list_length(this -> next);
```

```
};



bool list_equal(struct entry *this, struct entry *that){
    /*
    implementare in stile ricorsivo
    */

   if( this->value != that->value ){
    return false;
   }

    if( this == NULL || that == NULL ){
        if (this == that){
            return true;
        } else{
            return false;
        }

    }

    return list_equal(this->next, that->next);

};
```