SYSTEMS ARCHITECTURE

LAST NAME: .....
NAME: .....

LABORATORY SESSION 4

PROBLEM 1 (14th June 2016)
Sheldon Cooper loves order. So, he has decided (this is the nth time that he does something like this) to create several lists with all his comics, but now with a new brand classification.
In order to do so, he has defined a new data type:

```
struct comic{
    int number;
    char *collection;
    struct comic *next;
}
```

Moreover, he has created by hand the ordered list of all his comics, the header of the list is the variable `struct comic *all;`, defined in the code of the main function of the program that he is implementing (he dislikes profoundly global variables, as also does your professor). Also by hand, he created another ordered list `struct comic *DCcomics;`, with all the DC comics he owns (Batman, Superman, Green Lantern...).
Sheldon has asked (sorry, demanded) for our help in order to obtain the list of comics that **were not** published by DC Comics Inc (notice that the data type has no information about the editorial, and Sheldon does not want to add it, it would be an affront to his eidetic memory and, above all, to his ego).
Implement a function called `substract_lists,` that given 2 lists (listA and listB), creates and returns a new list, containing a copy of all the elements from listA that are not contained within listB.
This function should not modify neither listA nor listB. The elements should be sorted in the same order Sheldon used (Sheldon has stored them first alphabetically by collection, and for elements from the same collection he used the number of the issue, from lowest to higher).

This is the function that Sheldon, later, will use to obtain all his comics not published by DC, passing as parameters the list of all his comics and the list of his DC comics.

PROBLEM 2 (10th January 2020)
We want to build a circular linked list of keys. Thus, we have defined in C language
the following data type:

```
struct node{
    int key;
    struct nodo *next;
};
```

Implement the function add, that given the head of the circular linked list and a key, inserts the given key into the circular linked list. All the list should be allocated at Heap memory.
The function should not print anything, but it should return to the user if any error occurred during its execution.
It is not important where the new node is inserted in the list.
Remember that a circular list is a linear list in which the last node points to the first.

PROBLEM 3 (15th January 2015)
Kvothe is a very humble man, as he himself stated

"I have stolen princesses back from sleeping barrow kings. I burned down the town of Trebon. I have spent the night with Felurian and left with both my sanity and my life. I was expelled from the University at a younger age than most people are allowed in. I tread paths by moonlight that others fear to speak of during day. I have talked to Gods, loved women, and written songs to make the minstrels weep. You may have heard of me." (P. Rothfuss, The Name of the Wind)

However, he needs our help. He used to learn a lot of names (the real names of the things), and now he is forgetting them. . .

He knows a bit of programming, so he defined a dynamic array of structs. The definition of the struct is the following:

```
struct relationship{
    int id;
    char *real_name;
    char *common_name;
};
```

He even programmed the function `load`, that inserts into the dynamic array a new element (changing its size). However, he forgot to check if the new element was already in the array. So, right now, he has an array with a lot of duplicates and needs our help to *clean it*.

Section 3.1
Implement the function `delete_duplicates`, that given an array of the above defined data type and its size, deletes the duplicates within the array.
We will consider a duplicate any elements with the same `common_name`. The array does not follow any given order.
The function **should change the size of the array** and it should not have memory leaks. You can define any input and output parameters, but **you are not allowed to use global variables**.
You can also define new data types if you consider that it can help you (you should state why are you defining these new data types in your solution).

Section 3.2
Kvothe was thinking and decided that it is much better to use a linked list instead of an array. So, he defined a new data type:

```
struct node{
    int id;
    char *real_name;
    char *common_name;
    struct node *next;
};
```

Implement the function `create`, that given an identifier, a real name and a common name of a thing, creates and returns a variable of the previous data type.

The node and all the names should be dynamically allocated.

Section 3.3
Implement the function `copy` that given the head of a linked list, returns a copy of it (another linked list).
All the names and nodes should be dynamically allocated.

Section 3.4
Implement the function `mix` that given the head of **2 linked lists** `(oddList, evenList)`, creates a new list from their elements.
The function does not delete the previous lists and it should manage the unique identifiers of the new list. It will copy their elements in the following fashion:

- If any of the lists is empty, the function will copy the other one, and return the copy.
- If not, it should copy one element from the `oddList`, and then another from the `evenList` and so on, until it reaches the end of one (or both) of the lists.
- Once it reaches the end of one of the lists, if still there are elements in the other one, it should copy the remaining elements into the resulting list.

All the names and nodes should be dynamically allocated.