PROPOSED EXERCISES
Second day

Lecturer: Concha Batanero

The proposed exercises for the second day are also two. They are useful to apply the concepts explained in the slides. I recommend you to do first the function alluded on the number 25 slide. This will help you to a better control of the functions and their parameters in these proposed exercises. You must use the Memory Manager library and organize the code in several files.

1. Program to manage a group of monetary contributions

The owners of the apartments of an apartment block want to build a swimming pool in the garden. Every owner contributes with a specific amount of money to this end. Make a program that stores the quantities delivered by the owners of the apartments and displays the all quantities together with the total amount collected.

Take into account the next instructions:

The amount of money of every neighbour, will variate according with the number of meters of his or her apartment. So, the contribution of every neighbour will be read from keyboard and it will be a *float* type.

The number of neighbours of the apartment's block can be variable. Therefore, it must be initially read from keyboard.

Once all the contributions have been read and the results have been displayed, two new neighbours joint the group and make their contribution. The program must show the new result.

Finally, a neighbour located in a specific position of the list, which must be read from keyboard, decides unsubscribe from the list and recover his or her money. The program must manage this situation and show the final result.

You can download the executable file of the program to see the running of the application.

Program the next functions and organize the program in different files:

```
float * CreateArray(int nelem);
```

This function creates a dynamic array for the amounts of money. It receives as parameter the size of the array. It returns a pointer to the dynamic array.

```
float *AddElement(float *p, int *pnelem);
```

This function adds a new element to the dynamic array. It receives as parameters a pointer pointing to the beginning of the array and a pointer to the number of elements of the array. It returns a pointer to the dynamic array.

```
void DeleteElement(float *p, int *pnelem, int pos);
```

This function deletes an element of the dynamic array. It receives as parameters a pointer pointing to the beginning of the array, a pointer to the number of elements of the array and the position in the array of the element to be deleted.

```
void DisplayInfo(float * paux, int nelem);
```

This function displays every amount collected and the total amount once have been summed the all amounts. It receives as parameters a pointer pointing to the beginning of the array and the number of elements of the array.

2. Dynamic array of structures

Make a program that stores the name and the mark of a subject of a list of students. The program must read from keyboard the number of students and their data. After that, the program will display the information of the all students and will add a new student to the dynamic array in a specific position. Then will display the data again and will finish.

Think about the functions you need and their prototypes.