UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

# Systems Programming and Concurrency

# Lab nº3. Dynamic memory management and binary files

We want to generate a file with `SIZE` random values, each value being a number in the range between 0 and `SIZE-1`. To this end, we will first read `SIZE` from keyboard (the user will type the number), then we will generate the random values, and finally we will write them in a binary file, its name also read from keyboard.

Each random value is generated as the returned value of the rand() function. rand() will return a value in the range between 0 and `RAND_MAX`. To get a value in the correct range we can just use `rand()%SIZE` to obtain a value between 0 and `SIZE-1`.

To get different results every time we execute our code, it is necessary to seed with srand() the random number generator in a different way every time. A simple and practical way to do this is to use the current time, just like this: `srand (time(NULL));`. Then, we close the file and open it again; its content is read and displayed on the console; finally, we close the file again.

Again, we open the file one more time, discarding repeated values, sorting them from lowest to highest, and writing these ordered values again to the file. To discard repeated values and order them, we will use a binary search tree. Each node of the tree should have this type:

```
typedef struct TNode* TTree;
struct TNode {
     unsigned val;
     TTree left, right;
};
```

You are given a main() function. You have to finish the functions to read/write values from/to files and implement the following tree operations in `bst.c` (they are defined in `bst.h`):

```
// Create an empty tree
void create(TTree* tree);

// destroy the tree and free all memory
void destroy(TTree * tree);

// Insert the value in the tree. If it is already there, do
nothing
void insert(TTree* tree, unsigned val);

 // Show in screen the contents of the tree, ordered
 void show(TTree tree);

 // Save the tree to a file
 void save(TTree tree, FILE* f);
```