203-A01 More of the Handy Array Library

We are going to add functions to our Handy Arrays Library, using the files you already developed in 201-A01 to A04.

- **hal.h** will contain the headers (aka declarations) of our functions.
- **hal.c** will contain the definitions of all the functions we will implement.
- **tests.c** will contain the main function used to test our functions.

Work to do

To start this project, let's implement the following functions in hal.c;

- **int* intarray_allocate (int size);**
  This function returns the address of a newly allocated dynamical array of *size* integers or NULL if a *size* is negative or null or if *malloc* didn't work.

- **int intarray_deallocate (int * p);**
  This function will call *free* to deallocate the dynamical array pointed to by *p*. Make sure that *p* doesn't contains NULL since deallocating a NULL pointer would be an issue. If the deallocation went fine, return 0, otherwise return -1.

- **int intarray_deallocate_cleanly (int ** p);**
  This function will call the previous one on **p* and then will make sure that **p* is assigned NULL to prevent from using it erroneously. It will be called by using the address of a pointer on a dynamically allocated *int* array as effective parameter. Use the same return value convention than above

- **int* intarray_clone (int data[] , int size);**
  Allocates memory for a dynamical array of integers large enough to hold a copy of *data* then copies *int* per *int* the contents of *data* into the newly allocated array of integers and return its address or NULL if memory allocation didn't work

- **int* intarray_gather (int data1[], int size1, int data2[], int size2);**
  Allocates memory for a dynamical array of integers large enough to hold the contents of both *data1* and *data2* then copies the contents of both arrays, one after the other, in the newly allocated one and returns its address or NULL if the allocation didn't work.

- **int* intarray_extract (int data[] , int start , int end);**
  Allocates memory for a dynamical array of size (end-start +1) and copies inside the integers located between these indexes inside of *data*, you will then return the address of the newly allocated data or NULL if the allocation didn't work. You will assume the caller never provides a value for *end* which is greater or equal to the size of the array. You will also return NULL in the following cases;
    - data is NULL
    - start < 0
    - (end < start)

Testing

- Update *tests.c* to ensure that your functions are working. More specifically, make sure that you cover the error situations described above.
- Make also sure that you allocate a dynamical array and run it through all the tests you performed with a regular array before.
- How would you test for the following common pointer-related errors?
    - memory leaks?
    - free twice the same reference?
    - Dereference a NULL pointer
    - Dereference an invalid pointer
    - Any other pointer-related error you can think of?