More of the Handy String Library

We are going to add functions to our Handy String Library, using the files you already developed in 202-A05. Make sure you write the following functions by using strings features (e.g. end of string character) rather than attempting to reuse "as is" their counterparts from the Handy Arrays Library.

- **mystrings.h** will contain the headers (aka declarations) of our functions.
- **mystrings.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 mystrings.c;

- **char * str_allocate (int size) ;**
  - This function does the appropriate *malloc* to return the address of a newly allocated array of size+1 characters (to account for the additional '\0'
- **int str_deallocate (char* s);**
  - This functions operates as its counterpart in A01
- **int str_deallocate_cleanly(char* s);**
  - This function operates as its counterpart in A01
- **char * str_clone (char* s)**
  - allocates memory for a string to hold a copy of s
  - then copies char per char (including '\0') the contents of s into the new string
  - return its address or NULL if memory allocation didn't work
- **char* str_gather (char* s1 , char* s2)**
  - measures s1 and s2 lengths
  - allocate memory for a char * tmp to hold s1+s2
  - copy the strings in it
  - returns the address of this newly allocated string
- **char* str_extract (char* s1 , int start , int end);**
  - allocates a new string of length (end-start +1)
  - returns NULL if the allocation failed
  - returns NULL if s1 is NULL
  - returns NULL if (end > str_length(s1))
  - returns NULL if (end < start)
  - Then, copies in it the content of the string s1 located between positions start and end.

Testing

- Update *tests.c* to ensure that your functions are working (including in the error cases outline above).
- Make also sure that you allocate a dynamical string and run it through all the tests you applied previously to static strings.
- 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?