



## Lab1 – ADTs, dynamic memory, pointers & recursion.

**Important!** Read the document “Information and rules about the labs” . Read through the entire lab specs before starting to write code so that you get an idea of what to do, the extent and can plan your work and time. Read each assignment carefully: it is important that you follow the instructions to get everything done correctly. If something is unclear, contact your lab assistant: Gabriele Capannini ([gabriele.capannini@mdh.se](mailto:gabriele.capannini@mdh.se)) or Abu Naser Masud ([masud.abunaser@mdh.se](mailto:masud.abunaser@mdh.se)). You need to complete all the lab assignments to be eligible to attend the final lab examination.

---

### 1. ADT - Dstring

#### Task 1.1 Compile the skeleton

Download and extract the file `Lab1.zip` . Create a new project with a proper name and add all files from the skeleton to your project. Make sure you can compile the project without any error messages (there may be warnings: just ignore them at the moment). However, if you try to run the program, it will crash: it's normal at the moment.

#### Task 1.2 Get acquainted with ADTn DString

The `dstring.h` file contains the `DString` interface portion which is an ADT for dynamically allocated strings of text. It contains five function declarations. With these functions, you can: initiate/allocate, concatenate, truncate, write-to-file, and free-up the memory for dynamic strings. The file also contains comments describing what the functions do and some tips.

The `dstring_test.c` file contains an application that tests that the ADT works. Please carefully study this file and try to understand how the ADT is supposed to be used. When task 1.3 is complete, it should be possible to run this program without any errors.

The file `dstring.c` is unfinished. This will contain the implementation part of ADTs. It is your task to implement these functions. Before moving on to task 1.3, make sure you understand what all the functions in `dstring.h` should do. Please discuss with the lab assistant if you don't understand correctly.

#### Task 1.3 Implement DString

It is now time to implement the five `DString` functions in `dstring.c`. Set asserts for pre- and postconditions following the comments in the `.c` file that also contains some tips on

what to think about. Note that the type of `DString` is defined (via `typedef`) as a `char` array: i.e. a common string. Because arrays act as pointers, the type of `DString` type is `char*`. This means that the type `DString*` has the `char**` type, i.e., pointer to pointer/double pointer. This is necessary for those functions that change what the pointer (for example, when reallocating the dynamic memory).

Do not make any change to `dstring.h` or `dstring_test.c`, only to change `dstring.c` is allowed!

### Task 1.4 Testing DString

Now that you have implemented all the functions, it's time to see if your ADT works. Test is by means of `dstring_test.c`. Everything works well if everything compiles and the program ends normally. The program will also print the "[Hello World!][Hello]" on screen if everything works properly. If something else happens, you'll see where it's wrong.

**Good luck!**