aerospacelab

The linked list

1 Introduction

Let's assume you have been assigned to the software developement of sub-system of the next Aerospacelab Satellite. This sub-system is composed of a microcontroller which need to store items in a linked-list. Unfortunately, you have only access to pthread and the C standard libraries. You are asked to implement a **generic linked list**.

2 Implementation

The linked-list shall have the following API:

- `List_t *List_create(void)`: create a new List instance
- `int List_destroy(List_t *pList)`: destroy a List instance
- `int List_insert(List_t *pList, void *item, unsigned index)`: insert a new item at 'index' in the list
- `void* List_remove(List_t *pList, unsigned index)`: remove the item at 'index' from the list
- `void* List_get(List_t *pList, unsigned index)`: return the item at 'index' in the list (but don't remove it)
- `int List_push(List_t *pList, void *item)`: push (add) an item at the end of the list
- `void* List_pop(List_t *pList)`: pop (remove) the item at the end of the list and return it
- `int List_getSize(List_t *pList)`: return the number of item in the list
- `int List_sort(List_t *pList)`: sort all items in the list (consider the item pointer as an integer)

Note the following constraints:

- your implemenation shall be thread safe
- only pthread and the standard C libraries are allowed

3 Questions

- What algorithm did you implement to sort the list ?
- What is the complexity (big O) of the sorting operation ?
- What is the complexity (big O) of the insert operation ?

4 Delivery

We expect to receive:

- a C implemenation of the linked-list
- a small test suite to show your code is running as expected (you are allowed to use other libraries for this if needed)

- A Makefile or CMake to compile your code
- A brief rapport (PDF or Markdown) with:
  - the answers to the questions,
  - the choices you made and their motivations,
  - the encountered difficulties (if any),
  - a funny joke :-)

Tips:

- think about code readability (code format, comments, doc, ...)
- think about code testing
- all allocated memory must be free before end of the program
- organise your code in different files if needed