CPS 1000 Assignment

This assignment carries 40% of the final CPS 1000 grade.

Important Instructions:

You are to allocate *60 hours* for this assignment (excludes the development of lab exercises code fragments that are immediately reusable for this assignment).

Only students that regularly attend lab sessions are entitled to assistance from lab tutors. You will also be given continuous hints about how to carry out the assignment tasks during lectures.

The first page in your assignment report must be the title of your assignment clearly showing your name, surname and study unit code. The second page must the table of contents. The report sections should follow the tasks described in the following page.

While a fully functional software deliverable is a requirement the majority of the marks will be assigned for proper code explanations. Therefore the source code must be properly presented in the printed report along with the necessary code documentation/comments, as well as on disk (or other external storage). A *makefile* has to accompany the source code. Application(s) will be tested on Debian-based Linux or on Windows 8.1 (or later) according to the students' choice. A proper `readme.txt` file must be included in the disk's root folder and has to include: a description of disk content and instructions for setting up the application that includes the operating system on which it was tested.

PTO.

Title: The ‘priority queue’ abstract data type.

Context:

A definition for the `priority_queue` abstract data type (ADT) is given as:

Type name: `priority_queue`

Type properties: A collection of elements, where the dequeuing of elements is carried out according to an element's assigned priority. If two elements have the same priority they are served on a first-come-first serve basis.

Type operations:

- `create_q(`*`max_size`*`)`: create a new empty priority queue to store elements of a compound data type of your choice, having a maximum number of elements `max_size`.
- `enqueue(`*`Q,x,p`*`)`: enqueue element *`x`* to `Q`, with priority `p`. It is assumed that any individual queue is used to store elements of the same data type defined in `create_q()`.
- `dequeue(Q)`: dequeue element with highest priority from `Q`.
- `peek(`*`Q`*`)`: return, but do not dequeue element with highest priority from `Q`.
- `is_empty(`*`Q`*`)`: a predicate that evaluates to true if `Q` has 0 elements.
- `size(`*`Q`*`)`: returns the number of elements in `Q`.
- `merge(`*`Q,R`*`)`: create a new priority queue from the elements in `Q` and `R`, in this order, while retaining the same priorities of the input queues.
- `clear(`*`Q`*`)`: remove all elements of `Q`.
- `store(Q,`*`File`*`)`: store *`Q`* to disk in file *`File`*.
- `load(Q,`*`File`*`)`: load `Q` stored in file *`File`* to memory.

Tasks:

1. Research and then document a concise but illustrated description of the following data structures:
   a. Array
   b. Circular/ring buffer
   c. Linked list
   d. Binary heap **[10 marks]**
2. You are required to implement a library for the `priority_queue` ADT, taking care of good quality code in terms of comments, program structure, data type definitions, multiple source file organization and program robustness. The full source code, along with code fragment explanation is required to be included in the printed report. Specifically ***two versions*** of the library are required:
   a. An implementation that uses a circular/ring buffer. **[20 marks]**
   b. An implementation that uses a linked list. **[25 marks]**
3. Implement a single and basic/simple client application to test both library versions. **[15 marks]**
4. Test the functionality of the client application/libraries, using properly explained console sessions in your printed report. **[10 marks]**
5. Make sure your report is neatly formatted and that its content is legible. **[10 marks]**
6. Make sure that the software artefact(s) is fully functional. **[10 marks]**

Note: in subsequent study units of your course you will learn about more suitable data structures enabling for a more efficient implementation of this (and similar) ADTs. Knowledge of how to implement such data structures is a pre-requisite for those units.