COMP 2404/2004 – Project *cuTAES*

Project Activity: **Assignment #2**
Due: **Friday, February 15, 12:00 (noon)**
Collaboration: **You may work in teams of 2 or 3**

Background

You must be familiar with the *cuTAES* Project Description. Your code must include all working functionality from previous assignments.

Design Document

You will create a design document that contains the following:

- a UML class diagram that corresponds to your Assignment #2 program design; your diagram will depict:
  - all class names and class interfaces (public members)
  - all class relationships (composition and inheritance), with directionality and multiplicity
- a description and justification of your strategy for persistent data storage
  - describe what data must be saved in persistent storage between sessions, how the data is organized into separate files, and how the files relate to each other
  - justify why you chose your strategy, and explain how your approach promotes the principles of software engineering -- you must be specific!

Program Requirements

➢ **Feature functional requirements:**

- The *Create an application* feature for **multiple** applications by a student user must be implemented
  - the user selects a course for which to apply, from a preconfigured list of courses read from a file
  - the user is prompted for their *general information*, including student number, first and last names, email address, major, year standing, CGPA, and major GPA
  - the user is prompted for the information *specific to the course* for which they are applying, including:
    - a list of related courses taken (with term, year, and final grade)
    - a list of related courses for which the applicant served as TA (with term, year, and name of supervisor)
    - a list of relevant work experience (with responsibilities, duration, start and end dates)
  - all error checking must be performed on the data input
  - the user has the option to submit the application or cancel its submission
  - once the application is submitted, the user has the option of creating a new application for another course
  - if the user opts to create a new application:
    - they select a course for which to apply
    - a new application form is displayed, with the general information already filled in
    - they are prompted to enter the information specific to the new course for which they are applying
  - a user can create any number of applications for different courses
- The *View a summary of* pending *applications* feature for one selected course by an administrator must be implemented

➢ **Implementation functional requirements:**

- Storage of persistent data must be organized to suit the **complete** system, not just this assignment
- All collections used in the system must be implemented as a queue, represented as its own class
- The `Queue` class:
  - must be implemented as a **singly** linked list, developed from scratch, as covered in class and in the course notes
  - must include the following member functions
    - a default constructor, a copy constructor and a destructor
    - a `pushBack` function that adds an item to the back of the queue
    - a `popFront` function that removes the first item from the queue
    - a `front` function that returns the first item in the queue
    - an `empty` function that tests whether or not the queue is empty
  - your code must use the `Queue` class and its member functions wherever possible

➢ **Non-functional requirements:**

- The program must display the output in a graphical view, using either the Curses or GTK+ library
  - the program must not use stream functions for user I/O
- Design: the program must follow proper object-oriented design principles, including but not limited to:
  - encapsulation of data and functionality within classes, i.e. the principle of least privilege
  - easy modifiability and extensibility
  - no global variables and no global functions except `main`
- Implementation:
  - the program must explicitly deallocate all dynamically allocated memory
  - the program must not use prohibited library classes or functions (see Project Description for the list of prohibitions)
  - the program must not use any classes, containers or algorithms from the Standard Template Library (STL)
- Organization: the program must be correctly separated into source and header files
- Platform: the program must be written in C++, and it must compile and execute in the VM provided for the course

Submission

Each team will submit on *cuLearn*, before the due date and time, the following:

- a design document (as a PDF file)
- a C++ program that meets all the functional and non-functional requirements described above
  - all the program source code must be archived together into one `tar` file
  - the `tar` file must include a Makefile and readme file describing how to compile, execute and operate the program

Each individual team member will email comp2404@scs.carleton.ca before the due date and time, with the following:

- for each item in the marking breakdown, indicate the percentage of the submission contributed by each team member

Grading

- **Marking breakdown:**
  - Design document: 20%
  - *Create application* feature: 60%
  - *View summary of* pending *applications* feature: 20%

- **Deductions:**
  - Submission instructions or requirements not followed will result in severe deductions, including but not limited to:
    - **100 marks** if:
      - the assignment is submitted after the due date and time
      - any submitted files (including the Makefile) are missing, or corrupt, or in the wrong format
      - the code does not compile using g++ in the 32-bit VM provided for the course
      - the program does not use the Curses or GTK+ graphics library
      - the program uses stream functions for user I/O
    - **50 marks** if:
      - the design does not generally follow the principles of encapsulation and least privilege
      - global variables or global functions (other than `main`) are used
      - prohibited library classes or functions are used
    - **20 marks** if:
      - the submission consists of anything other than exactly one PDF file for the diagram and exactly one `tar` file
      - the program consistently crashes
      - the code is not correctly separated into header and source files
      - the readme file is missing or incomplete
    - **10 marks** for missing comments or other bad style (non-standard indentation, improper identifier names, etc)