Module:	ICP3038 – Computer Vision
Department:	School of Computer Science
Module credit:	20
Organiser:	Dr Franck P. Vidal
Assignment weight:	10%
Assignment deadline:	Friday 11th November 2016 (midnight)

PRIFYSGOL BANGOR UNIVERSITY

Assessment 1 – My own Image class in C++

Description

In this assignment, you will implement and test a C++ class to handle a 2D image. It is necessary to provide a good implementation of this class before coding any image processing. The implementation will be focusing on greyscale images stored as 1D arrays of single precision floating point numbers. The main reason is that it is the most versatile data type to store and manipulate pixel values. We provide a ZIP file that includes an initial skeleton (`Image.h`, `Image.cpp`, and `test.cpp`) and a file for CMake. You have to modify these files if needed to complete the assignment.

Contribution of this assessment

This assessment contributes to 10% of the overall module mark.

Tasks & Requirements

The assignment is based on the implementation of a C++ class for handling greyscale images. It should include:

1. Constructors:
   - `Image();` (default constructor)
   - `Image(const Image& anImage);` (copy constructor)
2. Destructor:
   - `~Image();` (destructor)
3. Assignment operator, also called copy operator:
   - `Image& operator=(const Image& anImage);` (assignment operator, also called copy operator)
4. Read/Write an image from/to an ASCII file:
   - `void readASCII(const std::string& aFileName)`
   - `void writeASCII(const std::string& aFileName) const`
5. Arithmetic operators using an another Image as a parameter:
   - `Image operator+(const Image& anImage)` (addition operator, pixelwise)
   - `Image operator-(const Image& anImage)` (subtraction operator, pixelwise)
6. Arithmetic assignment operators using an another Image as a parameter:
   - `Image& operator+=(const Image& anImage)` (addition assignment operator, pixelwise)
   - `Image& operator-=(const Image& anImage)` (subtraction assignment operator, pixelwise)
7. Arithmetic operators using a floating point number as a parameter:
   - `Image operator+(float aValue)` (addition operator, add aValue to every pixel)
   - `Image operator-(float aValue)` (subtraction operator, subtract aValue to every pixel)
   - `Image operator*(float aValue)` (multiplication operator, multiply every pixel by aValue)
   - `Image operator/(float aValue)` (division operator, divide every pixel by aValue)

8. Arithmetic assignment operators using a floating point number as a parameter:
   - `Image& operator+=(float aValue)` (addition assignment operator, add aValue to every pixel)
   - `Image& operator-=(float aValue)` (subtraction assignment operator, subtract aValue to every pixel)
   - `Image& operator*=(float aValue)` (multiplication assignment operator, multiply every pixel by aValue)
   - `Image& operator/=(float aValue)` (division assignment operator, divide every pixel by aValue)
9. Negation operator:
   - `Image operator!();` (return the negative image)
10. Accessors:
   - `unsigned int getWidth() const` (accessor on the width of the image)
   - `unsigned int getHeight() const` (accessor on the height of the image)
   - `double getAspectRatio() const` (method returning the aspect ratio)
11. Setters and Getters
   - `void setPixel(unsigned int i, unsigned int j, float v)` (method to set the value (v) of a pixel given its horizontal (i) and vertical (j) indices)
   - `float getPixel(unsigned int i, unsigned int j) const` (method returning the value of a pixel given its horizontal and vertical indices)
   - A method returning the smallest value in the image
   - A method returning the largest value in the image

Note: The functionalities of the class have to be tested and validated in `test.cpp`.

Task 1: Implementation

Your task is to complete the class to make sure every operator and method is implemented.

Task 2: Testing

The code has to be tested. Evidence of the results of the test will be assessed.

Task 3: Write a short logbook

Your final submissions should include the code and a logbook to provide evidence of testing. Discuss what works, and what does not. Whenever possible, use pictures to illustrate your text. At the end of your report, provide a critical analysis of your performance.
In appendix of your report, join your own copy of

- `Image.h`,
- `Image.cpp`, and
- `test.cpp`.

For testing, you can use ImageJ to load and save ASCII files (see `http://imagej.nih.gov/ij/download.html`).

Submission procedure

Write your report in a Word document. You must submit a .doc or .pdf file of your short report in the following format: "username-ass1.pdf". In appendix, include your source code.

The assignment is due on: Friday 11th November 2016 (midnight). Submissions must be made via Blackboard. Please take into account upload times and internet connections when considering how much time you have remaining.

Assessment method

Your report and your code will be marked. In addition, you will be interviewed during the one of the two labs following the deadline.

Note that care will be given to details. You are expected to use initialisation lists in constructors. No memory leak is allowed. Variable and function names should be meaningful. An appropriate coding standard should be used consistently. The code should compile without error. The code should compile without warning if possible. You should complete the header information at the top of each file.

Plagiarism and Unfair Practice

Plagiarised work will be given a mark of zero. Remember when you submit you agree to the standard agreement:

> This piece of work is a result of my own work except where it is a group assignment for which approved collaboration has been granted. Material from the work of others (from a book, a journal or the Web) used in this assignment has been acknowledged and quotations and paraphrasing suitably indicated. I appreciate that to imply that such work is mine, could lead to a nil mark, failing the module or being excluded from the University. I also testify that no substantial part of this work has been previously submitted for assessment.

Late Submission

Work submitted within one week of the stated deadline will be marked but the mark will be capped at 40%. A mark of 0% will be awarded for any work submitted 1 week after the deadline.

Acceptable reasons for submitting work late include: Serious personal illness with a doctors certificate (a self-certified medical note should not be accepted). The death of a relative or close friend. Serious family problems such as divorce, separation and eviction. Examples of unacceptable reasons for failing to submit work on time include: Having exams; Having other work to do; Not having access to a computer; Having computer related problems; Being on holiday; Not being able to find information about a subject.

Marking Scheme

Please remember that marks are provisional until they are confirmed by a board of examiners.

The marking will take into account:

- What is implemented, what is not implemented;
- The amount and usefulness of comments;
- The naming of functions, variables, etc. (including the consistency in the coding standard);
- Remove old code;
- Testing;
- Verbal explanation in the lab after the submission.

The marks will be split as follows:

1.	Constructors:	8%
2.	Destructor:	8%
3.	Assignment operator, also called copy operator:	8%
4.	Read/Write an image from/to an ASCII file:	14%
5.	Arithmetic operators using an another Image as a parameter:	7%
6.	Arithmetic assignment operators using an another Image as a parameter:	7%
7.	Arithmetic operators using a floating point number as a parameter:	7%
8.	Arithmetic assignment operators using a floating point number as a parameter:	7%
9.	Negation operator:	8%
10.	Accessors:	8%
11.	Setters and Getters:	18%

To understand the final grade, refer to the table below:

> 80, exceptional C++ programming skills. Clear demonstrable understanding use of C++ programming. Outstanding development of all the methods. Superb use of coding standards and comments. Overall an exceptional and well-designed test program that demonstrates the validity of every functionality. The report is exceptional and provides a comprehensive and clear critical analysis of the work performed. An exemplar solution that could be used to demonstrate good practice of parallel programming to colleagues.
> 70, good implementation that demonstrates a good understanding of C++ programming. The code works effectively and the test program is effective to check the validity of every functionality. The code is written using coding standards. The code is well commented. The report is comprehensive and makes a good critical analysis of the work. Overall a very good solution to this assignment. A well-structured report is provided (including a good critical analysis of the work provided).
> 60, a good implementation that demonstrates a suitable understanding of C++ programming. The code works effectively and the test checks the validity of most functionalities. Coding standards are relatively well applied and there are some comments. Some limitations may exist in the work, however a good attempt made, and although there may be some limitations with the program a comprehensive report (with an excellent and well critiqued section) is included.
> 50, appropriate demonstration of the challenge. Maybe full functionality is not provided, however the student demonstrates that they understand the processes required to achieve the assignment and understand some of the challenges and some of C++ programming.Comments are sparse and the use of coding standards is approximate. Even with these limitations the report is well presented, and their achievements are well criticised and discussed. Limitations to the work are clearly presented in the report and the student clearly understands what they have achieved and the limitations thereof. Overall, maybe some flaws, but a reasonable submission.
> 40, threshold performance. Demonstrates some understanding of C++ programming and some idea of testing the class. Some attempt has been made over creating the Image class. The code is not well commented and the coding standards are not used appropriately. The report discusses some of the issues, and provides a basic critique of the work submitted.
< 40, below threshold performance, with little demonstration of knowledge of C++ programming. Little thought has been made over this assessment, and understanding is confused.

Feedback details

	Description	Timeframe
Formative (On-going)	Verbal Feedback – Verbal feedback will be available by request. It is suggested that you keep a written note of this feedback to aid in your personal development.	Instant
Summative (Post Assessment)	Written Feedback – Written feedback will be made available through blackboard after an assignment is submitted. To access your written feedback see the comments section of your assignment submission.	1-2 weeks