New arrangements for COVID-19

As you are aware, there will be no in-house written exam for this module in Summer 2020. This is due to the national restrictions arising from the COVID-19 pandemic. We are mindful that you may have children at home, have sick or quarantining relatives, or may be in that situation yourself. We know that some have difficulties accessing the internet.

None the less, we want to provide you the opportunity to complete this module in the normal time frame. We have put in place an alternative assessment, involving a re-scoping of the project you worked on during the semester. The re-scoped project mark will now be used for the entire module.

By re-scoping, we mean that there are now four additional requirements for the project. Note that the work you have previously completed is still valid and will be assessed. The new requirements are as follows.

1. You must immediately make your GitHub repository private and add `ian.mcloughlin@gmit.ie` as a collaborator.

2. Your MD5 program must accept command line arguments. It should at least accept the `--help` command line argument, printing information about running the program. I recommend you include at least one other command line option, along the lines of the usual command line arguments accepted by standard Linux programs. You might use the `getopt` function from the C Standard Library for this.

3. You must include tests to check that the code is correct. These should run when the `--test` command line option is used.

4. You must add a file called `overview.md` to your repository. This Markdown document should contain a report on your project, pitched at students in the year below you. I recommend you consult GitHub's documentation on Markdown as to how to incorporate images, lists, links, and other items to your document. I recommend your document, when printed at normal size is at least eight pages long. Please include at least the following items in it.

   Introduction An introduction to your repository and code. Describe what is contained in the repository and what the code does.

   Run You should explain how to download, compile, and run your code. Include instructions of how to install the compiler.

   Test Explain how to run the tests included in your code.

Algorithm Give an overview and explanation of the main algorithm(s) in your code. You might use a well-thought out diagram here.

Complexity This should be the most significant part of the report. You must give an analysis of the complexity of the MD5 algorithm, including the complexity of algorithms that attempt to reverse the algorithm. That is, algorithms that attempt to find an input for which the MD5 algorithm produces a given output. You should research this topic before writing this section and your analysis should be carefully referenced.

References Provide a list of references used in your project. The references should not just be a list of websites. Instead, there should be a short explanation of why each reference is relevant to your document.

I appreciate it may take time to adjust to the ideas in this alternative assessment, and to this end the deadline has been pushed out to a last commit on or before April 30th, 2020. The original 30% will be marked as before, although you may use this opportunity to enhance that work by the new deadline. The following marking scheme will apply for the extra 70%.

15%	**Arguments**	Useful command line arguments informed by common command line tools in a typical Linux environment and as depicted by `--help`.
15%	**Testing**	Concise, well-documented, and effective tests of the algorithms.
40%	**Report**	Clear report explaining the technical aspects of the project. Language pitched at students in the year below. Typo-free and appropriately referenced.

Finally, we will work with you wherever possible to provide flexibility in achieving a fair and valid mark. It is always possible to defer assessment until the next sitting if there is documentary evidence that this is necessary. Unfortunately, it is not clear that the situation will have changed when the next sitting comes around in August. If possible, I recommend you try your best to complete this assessment now.

The rest of this document remain unchanged.

Project 2020

Theory of Algorithms

Due: last commit on or before April 30th, 2020

This document contains the instructions for Project 2020 for Theory of Algorithms. It involves writing a program in the C programming language [2] to perform the MD5 message-digest algorithm [5]. You will be required to demonstrate your program to the lecturer towards the end of the semester.

Please note that all students are bound by the Quality Assurance Framework [4] at GMIT which includes the Code of Student Conduct and the Policy on Plagiarism. The onus is on the student to ensure they do not, even inadvertently, break the rules. A clean and comprehensive git [1] history (see below) is the best way to demonstrate that your submission is your own work. It is, however, expected that you draw on works that are not your own and you should systematically reference those works to enhance your submission.

Problem statement

You must write a program in the C programming language [2] that calculates the MD5 hash digest of an input. The algorithm is specified in the Request For Comments 1321 document supplied by the Internet Engineering Task Force [5]. The only pre-requisite is that your program performs the algorithm — you are free to decide what input the algorithm should be performed on. I suggest you allow the user to input some free text or a filename via the command line.

Minimum Viable Project

The minimum standard for this project is a git [1] repository containing a single C file that calculates the MD5 digest of an input. The repository should also contain a README clearly documenting how to compile, run and test your program, how your program works, and how you wrote it.

A better project will be well organised and contain detailed explanations. The architecture of the system will be well conceived, and examples of running the program will be provided.

Submissions

The git software package [1] must be used to manage the development of your project and your git repository must be synced with an online provider such as GitHub [3]. Your repository will form the main submission of the project and will be submitted by providing the URL for your repository via the Moodle page. You can submit the URL at any time, the earlier the better, as the last commit before the deadline will be used as your final submission for the project.

Any submission that does not have a full and incremental git history with informative commit messages over the course of the project timeline will be accorded a proportionate mark. It is expected that your repository will have at least tens of commits, with each commit relating to a reasonably small unit of work. In the last week of term, or at any other time, you may be asked by the lecturer to explain the contents of your git repository. While it is encouraged that students will engage in peer learning, any unreferenced documentation and software that is contained in your submission must have been written by you. You can show this by having a long incremental commit history and by being able to explain your code.

Marking scheme

This project will be worth 30% of your mark for this module. The following marking scheme will be used to mark the project out of 100%. Students should note, however, that in certain circumstances the examiner's overall impression of the project may influence marks in each individual component.

25%	**Research**	Investigation of problem and possible solutions.
25%	**Development**	Clear architecture and well-written code.
25%	**Consistency**	Good planning and pragmatic attitude to work.
25%	**Documentation**	Detailed descriptions and explanations.

Advice for students

- Your git log history should be extensive. A reasonable unit of work for a single commit is a small function, or a handful of comments, or a

small change that fixes a bug. If you are well organised you will find it easier to determine the size of a reasonable commit, and it will show in your git history.

- Using information, code and data from outside sources is sometimes acceptable – so long as it is licensed to permit this, you clearly reference the source, and the overall project is substantially your own work. Using a source that does not meet these three conditions could jeopardise your mark.

- You must be able to explain your project during it, and after it. Bear this in mind when you are writing your README. If you had trouble understanding something in the first place, you will likely have trouble explaining it a couple of weeks later. Write a short explanation of it in your README, so that you can jog your memory later.

- Everyone is susceptible to procrastination and disorganisation. You are expected to be aware of this and take reasonable measures to avoid them. The best way to do this is to draw up an initial straight-forward project plan and keep it updated. You can show the examiner that you have done this in several ways. The easiest is to summarise the project plan in your README. Another way is to use a to-do list like GitHub Issues.

- Students have problems with projects from time to time. Some of these are unavoidable, such as external factors relating to family issues or illness. In such cases allowances can sometimes be made. Other problems are preventable, such as missing the submission deadline because you are having internet connectivity issues five minutes before it. Students should be able to show that up until an issue arose they had completed a reasonable and proportionate amount of work, and took reasonable steps to avoid preventable issues.

- Go easy on yourself – this is one project in one module. It will not define you or your life. A higher overall course mark should not be determined by a single project, but rather your performance in all your work in all your modules. Here, you are just trying to demonstrate to yourself, to the examiners, and to prospective future employers, that you can take a reasonably straight-forward problem and solve it within a few weeks.

References

[1] Software Freedom Conservancy. Git.
https://git-scm.com/.

[2] International Organization for Standardization. Iso/iec 9899 - programming languages - c.
http://www.open-std.org/jtc1/sc22/wg14/.

[3] Inc. GitHub. Github.
https://github.com/.

[4] GMIT. Quality assurance framework.
https://www.gmit.ie/general/quality-assurance-framework.

[5] Ron Rivest. Request for comments: 1321, the md5 message-digest algorithm.
https://tools.ietf.org/html/rfc1321.