Faculty of Technology – Course work Specification 2022/23

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Module name:** | | **Advanced Object-Oriented Programming in C++** | | | | | |
| **Module code:** | | **IMAT2906** | | | | | |
| **Title of the Assignment:** | | **Multi-Threading** | | | | | |
| **This coursework item is:** (delete as appropriate) | | | | Summative | |  | |
| **This summative coursework will be marked anonymously** | | | | | No | |  |
| **The learning outcomes that are assessed by this coursework are:**   1. Be able to synthesize a C++ OO software solution for a real-time simulation or game problem 2. Be able to critically evaluate how and when different OO techniques and design patterns should be used in order to solve problems typically found in software development 3. Be able to analyse the correctness and performance of C++ code using appropriate software tools | | | | | | | |
| **This coursework is:** (delete as appropriate) | | | Individual | | |  | |
|  | | | | | | | |
| **This coursework constitutes** 34% **of the overall module mark.** | | | | | | | |
| **Date Set:** | **5th July 2023** | | | | | | |
| **Date & Time Due:** | **15th August 12pm (Midday) 2023** | | | | | | |
| **The ‘normal’ coursework return date for this work is: 5th Spetember**  If for any reason this is not forthcoming by the due date your module leader will let you know why and when it can be expected. The Associate Professor of Student Experience (CEMstudentexperience@dmu.ac.uk) should be informed of any issues relating to the return of marked coursework and feedback.  Note that you should normally receive feedback on your coursework by **no later than 20 University working days after the formal hand-in date,** provided that you have met the submission deadline | | | | | | | |
| **When completed you are required to submit your coursework to:**   * Github Classroom repository   **If you need any support or advice on completing this coursework please visit the Student Matters tab on the Computing, Engineering and Media Blackboard page.** | | | | | | | |
| **Late submission of coursework policy:** Late submissions will be processed in accordance with current University regulations which state:  *“the time period during which a student may submit a piece of work late without authorisation and have the work capped at 40% if passed is* ***14 calendar days****. Work submitted unauthorised more than 14 calendar days after the original submission date will receive a mark of 0%. These regulations apply to a student’s first attempt at coursework. Work submitted late without authorisation which*  *constitutes reassessment of a previously failed piece of coursework will always receive a mark of 0%.”* | | | | | | | |
| **Academic Offences and Bad Academic Practices:**  These include plagiarism, cheating, collusion, copying work and reuse of your own work, poor referencing or the passing off of somebody else’s ideas as your own. If you are in any doubt about what constitutes an academic offence or bad academic practice you must check with your tutor.  Further information is available at:  [http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-](http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-offences.aspx) [offences.aspx](http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-offences.aspx) and  [http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-](http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-practice.aspx) | | | | | | | |

|  |  |
| --- | --- |
| [practice.aspx](http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-practice.aspx) | |
| **Tasks to be undertaken:**  Construct C++ code to demonstrate the following:   * Multiple threaded implementation that allows **at least** two differing operations (for example, process keyboard input such as WASD whilst displaying time to the console). * The use of a lock\_guard within a function. * The use of async and futures to read the .csv file provided in the GitHub repository and subsequently split the data so that only the column containing the game names is left.   Create a CMake file in order to construct the solution. | |
| **Deliverables to be submitted for assessment:**   * C++ code to GitHub classroom that contains all of the necessary files for it to run without changes. * CMake file to GitHub classroom. | |
| **How the work will be marked:**  Broadly your mark will be based on how well you have implemented the required elements, how much of the specification you have completed and how you have achieved it. Good code standards all improves your mark. You should make use of the techniques you have been taught to improve the efficiency, maintainability and reusability of your code.  The mark will be assessed using the attached marking rubric. It should be clear that these criteria are subjective and open to interpretation. To ensure a high mark you need to achieve good functionality with high quality code. Strong functionality will not make up for poorly hacked together code. High quality code will not make up for a lack of functionality.  If deemed by the module team, a number of individuals will go through a viva to discuss their code. This will not form part of your assessment but will explore any concerns the module team have about academic offences and bad academic practice. Ensure the work is your own – you may be selected for a viva and have to explain all aspects of your code. If you cannot explain your code you will receive a mark of zero and may be referred to the academic offences officer. | |
| **Module leader/tutor name:** | **Dr. Jethro Shell** |
| **Contact details:** | **jethros@dmu.ac.uk** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0-29% | 30-39% | 40-49% | 50-59% | 60-69% | 70-79% | 80-89% | 90-100% |
| **Implementation (75%)** | None / very few of the required tasks implemented. | Small number of the tasks implemented but there were issues / do not run and they do not correspond specifically to the requirements. | Small number of the tasks implemented. They run and correspond to the requirements. | Majority of the implementation completed. There are some bugs in the code and areas could have been extended. | Nearly a complete example of the required tasks. | Very nearly a complete example. All tasks are covered but there is scope for expansion. | An excellent example of the requirements. All of the elements are implemented. The software goes above and beyond the criteria set out. | Innovation. |
| **Structure (25%)** | No implementation or extremely poor structure to the code. | Poor structure to the code. Some misunderstanding of the subject that is being covered. | Some structure to the code, however no consideration for expansion, reusability or readability. | Reasonable structure to the code with the use of applicable commenting. | Strong structure to the code with some consideration of expansion, reusability and readability. | A very strong structure to the code with a clear understanding of the use of multithreads. Expansion, reusability and readability are all considered. | Thorough understanding of the subject which is demonstrated in the structure of the code, it goes above and beyond the specification. | Innovation. |