

School of Science and Technology

**COURSEWORK ASSESSMENT SPECIFICATION (UG)**

[Details of Module and Team](#module)

[What Learning Outcomes are assessed?](#module)

[What are my Deadlines and how much does this assessment contribute to my Module Grade?](#contribute)

[What am I required to do in the assessment?](#required)

[What are my assessment criteria? (What do I have to achieve for each grade?)](#criteria)

[Can I get formative feedback before submitting ? If so, how?](#formative)

[What extra support could I look for myself?](#extra)

[How and when do I submit this assessment?](#module)

[How and when will I get summative feedback?](#summative)

[What skills might this work evidence to employers?](#skills)

|  |  |
| --- | --- |
| **MODULE CODE** | SOFT10101 |
| **MODULE TITLE** | Computer Science Programming |
| **MODULE LEADER** | Amin Safaei |
| **TUTOR(S)** | Amin Safaei  Matthew Harris  Nuno Rodrigues Amalio  Alicia Falcon Caro  Sean Haddick  Michael Gibbs |
| **COURSEWORK TITLE** | Program Development |
| **LEARNING OUTCOMES**  **ASSESSED** | All |
| **CONTRIBUTION TO ELEMENT** | Part 2 of 2  70% of total module mark |
| **DATE SET** | Jan 2023 |
| **DATE OF SUBMISSION** | **14:30 on Sunday 17th April 2023**  **Demos in labs during *teaching weeks 25 and 26.) W/C 17th April and 24th April 2023.*** |
| **METHOD OF SUBMISSION** | Dropbox |
| **DATE OF FEEDBACK** | Three weeks from submission |
| **METHOD OF FEEDBACK** | Formative/Summative |
| Work handed in up to five working days late will be given a maximum Grade of Low Third whilst work that arrives more than five working days will be given a mark of zero.  Work will only be accepted beyond the five working day deadline if satisfactory evidence, for example, an NEC is provided. <https://www.ntu.ac.uk/studenthub/my-course/student-handbook/submit-a-notification-of-extenuating-circumstances>  The University views **plagiarism and collusion** as serious academic irregularities and there are a number of different penalties which may be applied to such offences. The [**Student Handbook**](http://www.ntu.ac.uk/current_students/resources/student_handbook/index.html) has a section on Academic Irregularities, which outlines the penalties and states that **plagiarism** includes:  'The incorporation of material (**including text, graph, diagrams, videos etc.**) derived from the work (published or unpublished) of another, by unacknowledged quotation, paraphrased imitation or other device in any work submitted for progression towards or for the completion of an award, which in any way suggests that it is the student's own original work. Such work may include printed material in textbooks, journals and material accessible electronically for example from web pages.'  Whereas **collusion** includes:  “Unauthorised and unacknowledged copying or use of material prepared by another person for use in submitted work. This may be with or without their consent or agreement to the copying or use of their work.”  If copied with the agreement of the other candidate both parties are considered guilty of Academic Irregularity.  Please remember submitting portions of work already assessed is **Self-Plagiarism** and is also a serious academic irregularity.  ………………………………………………………………………………………………………………………………………………….  **Information specific to incorporation of pre-existing program code.**  Whilst it is recognised that certain aspects of code can be re-used / adapted, such code MUST BE IDENTIFIED AND ACKNOWLEDGE WITH A REFERENCE.  You do not need to provide references for code reused/adapted from examples in the module learning materials, provided that you understand those examples. However, if you have reused/adapted a larger chunk of code from an external source ( for example open sources from the internet) then that external source should still be referenced.  **How to include references to program code**  Your references must identify precisely where the original code is to be found so that it can be examined when assessing your work. For example, a reference to a website must identify the specific webpage on which the code is found, whereas a reference to a textbook must identify the specific page(s) on which the code is found. If the precise location of the original code is not specified, then your work will be assessed under the assumption that the code was copied verbatim with no adaptations. For guidance on how to reference a webpage or textbook, see the 'NTU Guide to Citing References'.  Failure to identify and correctly reference reused/adapted code is **plagiarism** and will be processed under the school Suspected Academic Irregularity Policy.  …………………………………………………………………………………………………………………………………….  Penalties for Academic irregularities range from capped or zero grades for elements of modules, to dismissal from the course and termination of studies.  To ensure that you are not accused of plagiarism, look at the sections on [**Plagiarism Support and Turnitin support.**](https://www.ntu.ac.uk/m/library/plagiarism-and-turnitin) | |

**I.** **Assessment Requirements**

You will design and build a program in C++ along with a written report.

**Project Proposal**

You must write a program proposal as agreed with your lab tutor before starting and upload your proposal to dropbox by **19th  February 2023** (Project Idea for Approval) . The program must require a reasonable amount of design thought, without too much repetitive obvious coding, and exhibit skill in breaking down an apparently complex problem into individual elements. Those who did badly in the class test will be strongly advised to avoid too ambitious an assignment, as it may results in difficulties at a late stage. The degree of sophistication and intrinsic complexity in the problem will be taken into account during assessment. The program should relate directly to your course title in order to also act as portfolio or discussion topic with interviewers. (Your tutor may be able to suggest topics.)

**Program**

You must produce a working program with all source code, executable file and any required other files (eg images). You may use software from other sources for specific elements, but this software must be fully and correctly attributed in the program source files and in the declaration of ownership. The degree to which you have relied on existing code, and the ‘added value’ you have put in will form part of the assessment. Your ability to explain any code you have used in a professional manner will be included in the demonstration as part of the assessment. The software you produce must exhibit the following features where appropriate to the overall needs and design of the program. Features added purely to give evidence of that feature rather than as an intrinsic part of the design will have a negative effect! The features required are:

• Use of structs or classes

• Use of file handling

• Arrays or other collections of data items

**Report**

Because you are allowed some freedom as to what program you build, part of the task is to determine the detailed specification of your proposed product, and the acceptance tests that would show whether or not it achieved your goals. The report must be no more than 8 pages (font no smaller than 9 point). It must include sections on the following:

1. Specification of the proposed system breaking down into key features required.

2. Design/Implementation – description of the software elements that have to be written to achieve the system (this should include some diagrams eg flow charts, UML such as use case diagram, activity diagram, etc). Any particular implementation problems or issues should be commented on.

3. Test – details of tests that would confirm that the software performs according to the specification (including user faulty input where appropriate). NOTE this is not a transcript of the tests you have undertaken, but a specification of tests that could be applied to your program.

**Dropbox Submission**

You must submit the following:

1. Report as specified above

2. Declaration of ownership (which will include specific details of any code sourced from elsewhere)

3. Executable and source code, including .sln and .proj files (to enable the tutor to browse the project structure), and associated text, image files where necessary. Any code sourced from elsewhere must be clearly identified within the source files.

4. Demo File. A 5-10 minute demonstration of your program using video screen capture software.

\* The mark for this project (program development) will be mainly derived from a demonstration of the project which will include a question and answer session where your tutor will ask you about the design and structure of your project. Documentation will be used to help in this process. **If you do not attend your demonstration, or, the project shows inadequate ownership of the presented code, your tutor may award a mark of zero.**

**The Demo will be during lab sessions teaching week 25(w/c 17th April 2023) and week 26 (w/c 24th April 2023) according to your lab timetable.**

**II.** **Assessment Criteria**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class/grade  Assessment criteria | First | | | Upper second | | | Lower Second | | | Third | | | Fail | | | Zero |
| **\*Exceptional 1st** | | | High 2.1 | Mid 2.1 | Low 2.1 | High 2.2 | Mid 2.2 | Low 2.2 | High 3rd | Mid 3rd | Low 3rd | Marginal fail | \*Mid Fail | \*Low Fail | zero |
| High 1st | Mid 1st | Low 1st |
| **Functionality (55%):** | As previous plus extra features as agreed beforehand with tutor which use highly sophisticated and considered language constructs, and achieves considerable functionality or difficult goals, that will have required the use of considerable design skills. Clear reflective thought on design, and pertinent testing. Can discuss code and rationale highly professionally, with consideration of alternatives, design and implementation issues etc.  \*Exceptional programming skills and coding quality. Program satisfies specification completely and correctly. | | | Working program that achieves major requirements and substantial content. Program shows use of extra features appropriately deployed and performs a task of some complexity, requiring good design. | | | Working program that has basic interface handling, but can only achieve partial success in key requirements. A reasonable amount of code, though size may be gained from repetition rather than novel sections. Includes some features beyond those in live test. Program performs a reasonable task of moderate intrinsic complexity. | | | Working program that has interface handling but cannot perform much functionality, or task relatively simplistic, and requiring very little design. Limited understanding of code or code so simple that needs little understanding. Cannot explain how additional features might be added. | | | A small part of program works but main functionality of the program is not working correctly.  \*Program that does not satisfy any key requirements. Program shows lower level of skills than those used within the live test. Understanding of code minimal - explanation of even simple aspects unsatisfactory | | | Work of no merit.  Absent from test |
| **Object-Oriented Programming (20%)** | Excellent object-oriented design and well-thought out implementation, using sophisticated constructs where appropriate. Shows effort/learning well beyond taught material. Program succeeds completely at achieving its aims, or approaches very challenging aims very well  \*Exceptional object-oriented design which uses of most object-oriented programming concept that binds together the data and functions that manipulate the data and uses polymorphism. | | | Very good Understanding of how to identify, illustrate and describe all the classes in the program. All classes are accurately and completely discovered and implemented. | | | Good understanding of how to identify, illustrate and describe all the classes in the program. | | | Sufficient understanding of object-oriented modelling but the implementation is not complete. Many classes and their characteristics are missed or, not implemented correctly including some major ones. | | | Marginal understanding of object-oriented implementation.  \*Insufficient understanding of object-oriented implementation. | | |  |
| **Modularity (10%)** | Program is decomposed into coherent and reusable units, and unnecessary repetition has been eliminated.  \*Exceptional modularity which uses header files, use of encapsulation. Program is fully decomposed into coherent and reusable units. Ability to decompose a problem into coherent and reusable functions, files, classes, or objects (as appropriate for the programming language and platform). | | | Program is decomposed into coherent units, but may still contain some unnecessary repetition | | | Program is decomposed into units of appropriate size, but they lack coherence or reusability. Program contains unnecessary repetition | | | Program is one big function or is decomposed in ways that make little sense. | | | Marginal understanding of modularity.  \*Insufficient understanding of modularity. | | |  |
| **Comments and Documentation (5%)** | Very good documentation and comments. Well-formatted, understandable code; appropriate use of language capabilities  \*Exceptional ability to format and document code for human consumption. Program contains appropriate documentation for all major functions, variables, or nontrivial algorithms. Formatting, indentation, and other white space aids readability. | | | Very Good documentation and comments. Understandable code. Report shows clear process from requirements, through design decisions to quality testing. Discussion of code is good and competent, with understanding of its limitations. | | | Good documentation and code comments, but the code is hard to follow in one reading; poor use of language capabilities. Discussion of code has some understanding, but may be missing in depth or consideration of limitations of particular choices or structure. | | | Sufficient documentation and a reasonable use of comments and code | | | Incomprehensive code, appropriate language capabilities not used. The code has no comments.  \*No code comments No program documentation | | |  |
| **Testing (10%)** | Excellent testing strategy and excellent use of testing  \*Exceptional testing strategy and use of testing tools | | | Very good testing covering all functional and non-functional requirement. | | | Good use of testing (test plan) | | | Sufficient use of testing. | | | Marginal understanding of testing | | |  |
| **Comments** | | | | | | | | | | | | | | | | |

**III. Feedback Opportunities**

**Formative (Whilst you’re working on the coursework)**

You will be given the opportunity in Labs to discuss your coursework with your tutor.

**Summative (After you’ve submitted the coursework)**

You will receive specific feedback regarding your coursework submission together with your awarded grade when it is returned to you. Clearly, feedback provided with your coursework is only for developmental purposes so that you can improve for the next assessment or subject-related module.

**IV. Resources that may be useful**

Referencing styles please use Harvard as detailed [here](https://www4.ntu.ac.uk/library/developing_skills/referencing_plagiarism/referencing_styles/index.html)

Guide to planning your time [here](https://www.kent.ac.uk/ai/ask/index.php) and an automated planner [here](https://now.ntu.ac.uk/d2l/le/content/52836/Home?itemIdentifier=D2L.LE.Content.ContentObject.ModuleCO-1577785)

Further guidance on avoiding cheating is [here](http://www4.ntu.ac.uk/current_students/studying/skills-for-success/copyright-plagiarism/plagiarism.html)

Remember to use Outlook or physical calendars to block out time between lectures and labs to work on this coursework.

**V. Moderation**

**The Moderation Process**

All assessments are subject to a two-stage moderation process. Firstly, any details related to the assessment (e.g., clarity of information and the assessment criteria) are considered by an independent person (usually a member of the module team). Secondly, the grades awarded are considered by the module team to check for consistency and fairness across the cohort for the piece of work submitted.

**VI. Aspects for Professional Development**

|  |  |
| --- | --- |
| **BEST image** | * BEST skills |
| Icon  Description automatically generated with medium confidence | **Professional Practice**   * Understand more about the demands of a professional work context and possible scenarios * Learn how constraints and requirements affect working practice (quality, time, and documentation standards *etc*.) |

This assessment cover examples of

Writing a scientific-style report; Researching existing literature; Referencing appropriately; Construction of efficient programs; Construction and proper labelling of figures; Critical evaluation of methods

Many of these are useful transferable skills for employment applications. A successful product can be part of your portfolio of work, which is useful as a discussion topic with interviewers when applying for placement.