

In-Course Assessment Brief

Undergraduate Programme Academic Year 2009/2010

Module:	Data Structures and Algorithms UG2
Assessment Title:	Stock control database program.
Assessment Identifier:	CWK1
School:	Computing, Telecommunications and Networks
Module Co-ordinator:	Richard Kay
Assessment Details and Deadlines:	See ECMS My Course on the intranet.
Brief Assessment Details	Students will study and reverse engineer a stock control database program downloaded from the module website. Students will develop documentation and source code in 'C' to describe, test and implement the functionality of the program as provided..

If you should fail this module you will be permitted to be re-assessed on up to three occasions. If you fail to attend or to submit work for re-assessment at the next opportunity you will be deemed to have exhausted one of the opportunities.

IMPORTANT STATEMENT

Plagiarism: the presentation of the work of another (from whatever source: book, journal, internet etc) as if it were one's own independent work. This can be anywhere on a continuum ranging from sloppy paraphrasing to verbatim transcription without crediting sources.

You are advised to refer to the Student Handbook on matters of cheating and plagiarism as they relate to coursework, group assignments, class tests and examinations. Both cheating and plagiarism are totally unacceptable and the University maintains a strict policy against them. It is YOUR responsibility to be aware of this policy and to act accordingly.

The University requires that the following statement is included in all module documents.

"You are reminded of the University Disciplinary Procedures which refer to cheating. Except where the assessment of an assignment is group-based, the final piece of work which is submitted must be your own work. Close similarity between assignments is likely to lead to an investigation for cheating. It is not advisable to show your completed work to your colleagues or to share and exchange disks.

You must also ensure that you acknowledge all sources you have used. Work which is discovered to be the result of collusion or plagiarism will be dealt with under the University's Disciplinary Procedures, and the penalty may involve the loss of academic credits.

If you have any doubts about the extent to which you are allowed to collaborate with your colleagues, or the conventions for acknowledging the source you have used, you should first of all consult module documentation and, if still unclear, your module tutor."

You will be asked to confirm in writing when handing in any piece of assessed work that it is your own by completing the Coursework Submission & Record Form which should be printed from ECMS My-course on <https://mytid.bcu.ac.uk/>.

It is the STUDENT'S responsibility to accurately complete the form and comply with its rules and guidance as described in the student handbook for this academic year.

Learning Outcomes to be Assessed:

Demonstrate an understanding of the application and use of pointers in data manipulation. Demonstrate an understanding in the skills required to manipulate data at various levels. Apply acquired knowledge and understanding to implement simple data structures and algorithms.

Assessment Details:

For full details see also module website: <http://bcu.copsewood.net/tic/dsalg>

Students will work either singly or in teams of 2. Where students work in teams of 2, only the source code will be marked as a team effort, and all other deliverables require individual and entirely independent student work. Names of both team members must appear on all coursework submissions and in source code comments. Submissions of design and test documentation and reports must be individual work for all students, and in situations where evidence exists e.g. from widely differing quality of individual submissions, that unequal work was done by different team members, individual marks for the team deliverable will be moderated appropriately. Students submitting work done as individuals must ensure they hand in only work done independently by themselves and that none of this work is shared with or contributed to by anyone else. If this is not done, any commonality of work handed in will be considered evidence of derivative work and substantially marked down, and in serious cases may result in academic cheating allegations being pursued.

A binary program and an associated data file has been made available for download from the module website. Students are required to study the operations of this program carefully in connection with a simulation of a set of stock control operations. Having done this, students are required to design, implement using ANSI 'C', and test an equivalent program. Other documentation and reports must be included in coursework handed in as detailed below. Teams of 2 students working on this assignment are required to complete additional tasks and submit additional deliverables. Students are not allowed to work on this assignment in teams of more than 2 members.

Deliverables

Design documentation (individual deliverable)

Each student will individually document, in their own words, the program specification. Diagrams may be used to describe inputs and outputs etc. Students will also document the design of their equivalent programs, describing for each program module or global section, input, output, relevant data variables, constants and data structures and relevant algorithms using pseudocode or a flowchart or other suitable means.

Test documentation (individual deliverable)

Each student will individually document a set of tests comprising a structured plan which must be sufficient to enable an independent test engineer to prove the correct program operations using identical input as detailed in the test plan to produce identical output as

detailed in the test plan. The test plan should comprise a set of test cases, with each case testing a discrete aspect of program functionality. For each test case:

- a. the purpose and content of the test will be described,
- b. any prior actions needed,
- c. the actual live input to be used for the test/s and
- d. the observable output value/s expected, and
- e. any non-obvious steps needed to make relevant observations should be stated.

Screen shots of selected significant program outputs may be included in the report as supporting evidence of successful test outputs.

Program source code (individual or team deliverable)

This is either an individual or a team submission. For team submissions marking will be moderated based on any evidence of unequal team effort. Both printed and electronic format (CD or floppy) are required. This should be suitably commented and should be appropriately indented and modularised. Appropriate variable, function and constant names should be used. Individuals working on their own must replicate all facilities in the provided binary program except for features described under additional team tasks below.

Conclusion reports (individual deliverable)

Each student must independently write a conclusions report (suggested length 300 - 400 words). This should describe the development process undertaken, observations made and lessons learned. Where an individual has worked as part of a team, an honest individual assessment must be included describing team decisions and interactions. Recommendations should be made covering any improvements which could be introduced in connection with individual (and where appropriate team) development processes if a similar exercise were to be attempted again.

Additional team tasks (teams only - or individuals with time available seeking to achieve and demonstrate excellence in completion of all learning outcomes).

Students undertaking this assignment in teams of 2 or working individually in pursuit of good or excellent outcomes, will, in addition to all the other facilities present in the downloaded programs, implement program facilities to enable records to be modified and deleted from the database. These changes must be persistent through program ending and restarting using the external data file output and input. Students undertaking this assignment in teams of 2 will also implement a facility to sort the database records, in ascending order keyed on a user-selected choice of 2 columns: either the Part Number, or the Number in Stock. If a student working individually attempts and completes some or all of the above team tasks and consequently demonstrates greater completion of learning outcomes this might enable minor gaps in other deliverables to be moderated accordingly; however, this compensation can only be given if the student has clearly demonstrated adequate attention to completing all assigned individual tasks first.

Assessment Criteria:

See assessment and grading criteria table below.

Table of Assessment Criteria and Associated Grading Criteria

Assessment Criteria →	1. Design Documentation	2. Program source code quality and performance	3. Test plan and test report	4. Conclusions
Weighting :	20%	50%	20%	10%
Grading Criteria 0 – 29%	Separate design document not done. Possibly a few comments in source code	Very poor quality source code. Little or no programming work done.	No test plan or results or trivial approach.	Conclusions not written or trivial and very weak. No evidence of thought concerning project.
30 – 39%	Weak effort lacking modularity demonstrating little understanding of task or program.	Parts of program implemented too small to pass.	Few tests planned and documented. Little test coverage.	Weak report not demonstrating much evidence of thought applied to the project.
40 – 49%	Some documentation relevant to program but of poor standard and lacking full understanding of task.	Enough of program implemented to pass but very large gaps and code of limited quality.	Enough planned test coverage to pass but with large gaps.	Passable report but only just. Some awareness of development process but this is very limited.
50 – 59%	Better understanding of program and task but significant gaps.	Core subset of program implemented to a fair standard but with gaps.	Fair set of tests planned and recorded but limited and unsuitable for independent testing.	Some thought about development process but with large gaps and lacking a fully critical evaluation.
60 – 69%	Good documentation of program but with minor deficiencies.	Implementation of nearly all program features but with bugs or a questionable approach to coding.	Good set of tests adequate to enable another person to test program, but with some deficiencies.	Thoughtful and critical review of development process but with some shortcomings.
70+%	Full documentation of program describing a well-designed program completed to a high standard.	Complete implementation to a high quality with good programming style and modularity.	Full set of independently repeatable and verifiable tests and results very clearly presented.	Fully self-aware, authentic and deeply thoughtful report demonstrating an excellent standard of observation.