Faculty of Technology – Course work Specification 2016/17

Module name:	Computer Programming		
Module code:	CTEC1403		
Title of the Assignment:	Assignment 4		
This coursework item is: (delete as appropriate)		Summative	
This summative coursework will be marked anonymously		Yes	
The learning outcomes that are assessed by this coursework are: 1. Analyse a problem and produce a program specification 2. Design a program to a given specification using the control and data structure abstractions provided by a contemporary programming language 3. Deploy trusted software design techniques in the construction of a computer program 4. Apply relevant software testing techniques to verify the components of a computer program			
This coursework is: (delete as appropriate)		Individual	
This coursework constitutes 25% **to the overall module mark.**			
Date Set:	Week 22		
Date & Time Due:	Friday 20th April 2018 (Week 29) - 23:59 (Blackboard time)		
Your marked coursework and feedback will be available to you on: 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 Head of Studies (headofstudies-tec@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 four working weeks after the formal hand-in date,** provided that you met the submission deadline.			20 working days from due date.
When completed you are required to submit your coursework to: • Blackboard VLE through an assignment submission portal			
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% [50% at PG level] 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 and details of how DSU can support you, if needed, is available at: 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-practice.aspx			
Tasks to be undertaken: See (following) attached document.			

Deliverables to be submitted for assessment:

You are required to complete the implementation of the Buffer (editor) library: Buffer.scala. The specification of each of the operations is provided as comments within the skeleton code. You may complete as many of the operations as you wish.

You will be provided with a JUnit test case in which there are 100 unit tests. You can use this to test your functions incrementally. This approach to software construction is called *test-driven development.*

You should upload to Blackboard the Buffer.scala file – nothing else is required.

Please put your P-Number at the top of the Scala file as a comment, but not your name.

How the work will be marked:

The markers will run your library against the published JUnit tests and allocate a mark based upon how many tests are passed. Therefore, normally, the greater the number of operations you complete successfully, the higher the mark will be. You do not need to hand in the JUnit test cases because we will be using the original JUnit test case file (i.e. BufferTest.scala).

Note that initially some of the unit tests already pass even with no implementation. This is because they are testing, e.g., certain boundary conditions which are initially true. However, there is no guarantee that these tests will still pass once you start coding them. They will of course if you code them correctly.

The markers will view your source code to check that you have made a reasonable attempt at the operations and reserve the right to adjust the final mark accordingly. (Therefore, leaving all the methods as unimplemented in order to pick up free marks will not be a successful strategy!)

Module leader/tutor name:	**David Smallwood**
Contact details:	drs@cmu.ac.uk (GH6.71)

Important note

Please do **NOT** be tempted to search the internet for an existing solution and then hand this in – e.g. if you run out of time. This is cheating – it is better to hand in what you have honestly achieved by yourself than try to get credit for someone else's work and risk getting caught. Cheating is an **academic offence**.

If you get stuck then please ask the module tutors for assistance and come to the labs – we will be providing help in the labs (but not actually doing it for you of course).

Do **NOT** use anyone else's material without referencing it – this is **bad academic practice** or **plagiarism**. These are considered as **academic offences**.

Do **NOT** work jointly on a solution; do **NOT** give your solution to anyone else to "help them" and do **NOT** accept anyone else's solution as "guidance". Such practice could lead to an allegation of **collusion** which is an **academic offence**. **All parties** involved in collusion (the givers, the receivers, the collaborators) can be found guilty of an academic offence, irrespective of motive.