e-Mark

An Excel program for automatic marking of assignments

Dr Karen Ayres

The University of Reading

January 2009

Table of Contents

Acknowledgements 3
1. Introduction 4
   1.1 Automatic marking 4
   1.2 Features of e-Mark 5
   1.3 Appearance 6
   1.4 Office 2000, 2003 or 2007 9
   1.5 Before you start 9
   1.6 Overview 10
2. Dataset provision 11
   2.1 Entering datasets 12
   2.2 Output format 15
   2.3 Protecting the file 16
   2.4 Possible problems 18
3. e-Mark – initial set-up 19
   3.1 Introduction 19
   3.2 Students 19
   3.3 Data 20
   3.4 Student solutions – Word or Excel? 20
   3.5 Solutions, marks and tolerance 21
   3.6 Consistent errors 23
   3.7 Common errors and feedback 25
4. e-Mark – running the program 27
   4.1 Introduction 27
   4.2 Possible problems 27
   4.3 Marking the work, with feedback 27
   4.4 Auditing the work and human intervention 30
   4.5 Summaries of solutions and marks 31
5. Word file for student solutions 33
   5.1 Introduction 33
   5.2 Generating files for handing in solutions 33
   5.3 Processing the student files 35
   5.4 Generating files for returning marked work 36
6. Problems and advice 40
7. Possible future changes 42
Appendix A – Datagen code 43
Appendix B – e-Mark code 53
Appendix C – Master code 76

Acknowledgements

The e-Mark suite of programs was developed in 2008 at the University of Reading by staff in Applied Statistics. All code is written in Visual Basic in Microsoft Excel (or Word).

The code for each of the programs described here was written by Dr Karen Ayres, Executive Director of Teaching & Learning and Lecturer in Applied Statistics, as part of an e-learning project at the University of Reading in 2008-09. This project was supported by a grant of £700 to Karen Ayres, Dr Fiona Underwood and Dr Mike Dennett from the University's Centre for Development of Teaching and Learning. The programs were implemented and evaluated by Fiona Underwood, who made significant use of the program and highlighted many important areas for improvement, and also by Mike Dennett. Alex Owen created the Word files for student solutions and feedback for the courses that made use of the e-Mark program.

Address for correspondence:
Dr Karen Ayres
Applied Statistics
School of Biological Sciences
The University of Reading
Philip Lyle Building
Reading
RG6 6BX
E-mail: k.l.ayres@reading.ac.uk

1. Introduction

1.1 Automatic marking

It is clear that students benefit from timely feedback, and also from regular assessments that allow them to engage in active learning. However, with large classes it becomes more difficult to mark assignments in a reasonable timeframe, without resorting to simple ticks and crosses and adding no further feedback. It is possible that employing some kind of computer-based marking of assessments will allow for a quicker completion of the marking process, and allow for provision for more detailed feedback where obvious errors can be foreseen. Quicker feedback obviously allows for better self-reflection by students with regard to their strengths and weaknesses, thus aiding their academic development.

As well as quicker feedback, students will be able to benefit in further ways. First, e-assessment is open to the problem of plagiarism unless carried out in a controlled environment. With automatic marking performed by computer, it is no more time-consuming for an academic to provide students with individualised data relating to quantitative or qualitative questions, than to provide them with the same data (beyond the initial set-up costs of generating different data that is). So, individualised data is very much a part of this program. Such a facility helps discourage cheating, as straight copying of solutions is no longer possible. At the very least, those students who don't wish to attempt the questions on their own will have to consult others on how to answer the question, and (presumably) go through the steps of applying the method to their data. This opens up the possibilities for peer-group learning, which can be a very effective method for enhancing understanding.

Furthermore, linked to the University's interest in e-enhancement of learning and teaching, although the program is primarily intended as an auto-marking framework for summative assessments, the material has the potential to be developed for self-study and revision purposes. For example, it would be easy to provide a range of simple quantitative questions plus several possible datasets within an Excel file, and allow the student to attempt the questions in their own time and mark them themselves at a click of a button, using the code provided. This would be particularly useful for those students who appreciate the opportunity for practising methods repeatedly, and ultimately may be useful for enhancing numeracy skills.

From the lecturer's point of view, having the potential to mark a large volume of simple assignments in a timelier manner is of benefit, since most have a heavy workload. The issue with having multiple sets of datasets and solutions is not one created particularly by using an auto-marking program, since copying is often present anyway, and there may already be the desire to provide individualised questions/data to students to prevent this. The program makes this more feasible. However, set-up costs should not be overlooked, because as well as the time taken to enter solutions into the program, additional time is needed in advance to determine marking schemes for partially correct solutions, and appropriate feedback for common errors. Much of this latter work would usually be done at the time when marking took place by hand, but when using e-Mark it needs to be thought about carefully beforehand. However, a positive is that once common errors have been specified in the program, they can be detected, along with errors that have been carried through to later answers, meaning that the lecturer does not have to work things through by hand to find out what has gone wrong. This can be a great time saver when the program is used instead of marking by hand.

A potential negative of an auto-marking program would be that it cannot mark graphs, discursive passages, nor theory and proof. However, provision is made to allow for additional components of

assignments to be marked external to the program, and marks entered in manually before the final feedback sheets are produced.

1.2 Features of e-Mark

In a nutshell, the e-Mark program is a set of Excel macros (written in Visual Basic) that are designed to be generic, which

- collate all student answers into a single Excel worksheet
- check all student answers against the actual answers
- annotate student answer sheets with appropriate pre-determined feedback
- provide an overall summary for the academic, including marks

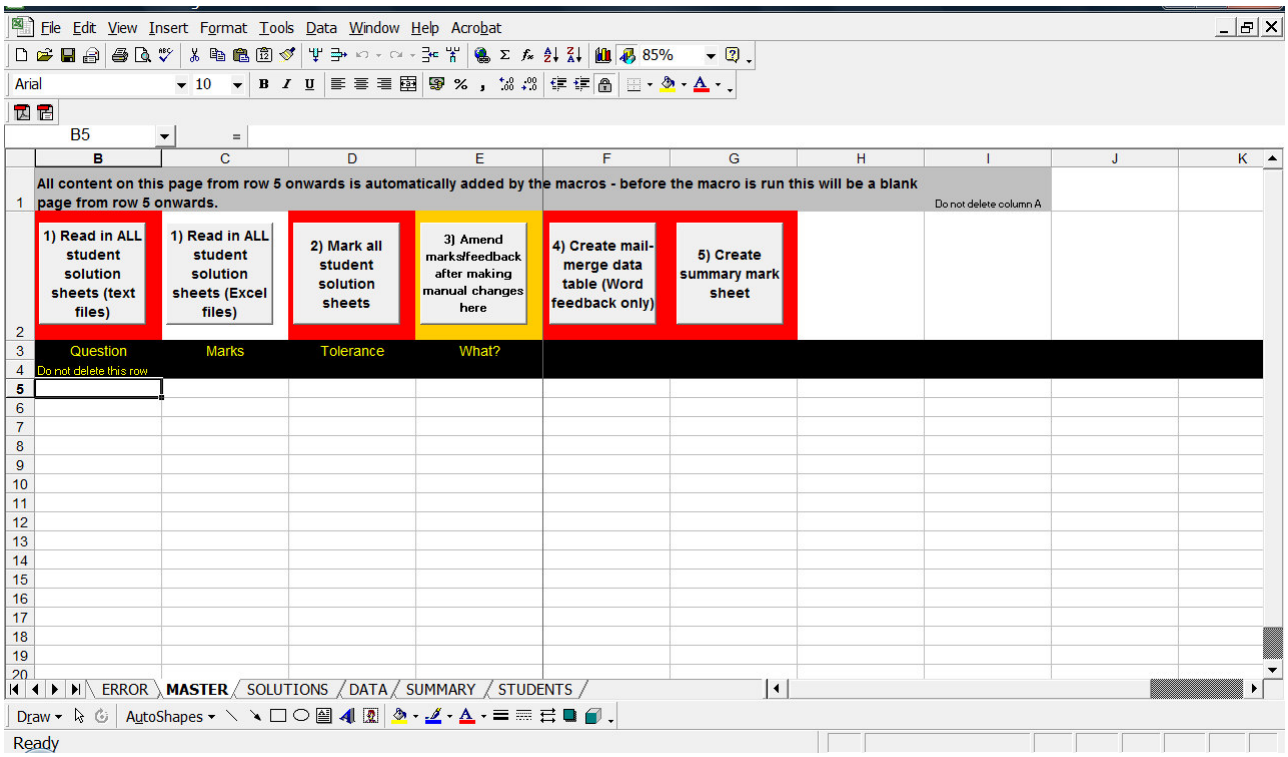
The particular features of e-Mark are listed below. These are described in more detail later in this document.

- Allows for a different set of solutions to be allocated to each student (e.g. due to different datasets).
- Can accept student solutions provided in Excel or Word format (Word format must be saved as text files first), thereby allowing flexibility with the format of the solution and feedback sheets provided to students.
- Allows flagging of students with special needs, to provide a note that work has been marked in accordance with this.
- Numeric and pre-specified text solutions can be marked (i.e. from multiple choice).
- Numerical solutions can be marked to within a specified tolerance, allowing for e.g. slight rounding error.
- Numerical solutions can be checked for whether they are correct given that a fewer number of decimal places has been used when entering the solution than desired.
- Provides ticks/crosses as well as specified feedback for incorrect solutions.
- Checks any solution for an error caused by carrying forward an earlier error, thereby allowing marks to be given for the method used, i.e. a “consistent” error.
- Allows forcing of consistent error checks, to prevent a lucky guess from being marked correctly when it does not match the earlier incorrect answer on which it is based.
- Allows any number of foreseen “common” errors to be specified with feedback for each question, to help identify where a student has gone wrong.
- Common errors can include a check of validity of numerical solutions, i.e. whether less than the theoretical minimum, or greater than the theoretical maximum.
- All consistent and common error checks can be specified by the user as standard Excel functions, which allows for flexibility and means that no specific knowledge of Visual Basic is needed. Use can be made of Excel’s mathematical and statistical functions, for example, as well as standard arithmetic on single or multiple cell entries.
- All student sheets are read into the program at the click of one button.
- All student solutions are marked at the click of one button.
- All student solutions are copied to one Excel sheet, and are shown together with the real solution, the feedback provided, and the mark awarded.
- Student solutions are colour-coded on the summary sheet, to allow easy identification of type of error made.
- Allows manual updating of marks and feedback after auditing the program’s results.
- Allows manual addition of marks and feedback for e.g. graphs, theory, discussion that must be marked by the lecturer outside of the program.
- Allows manual addition of one or two pieces of overall feedback.

Note that some other programs, e.g. Blackboard, allow for marking of solutions, including a tolerance. However, it is usually not possible to allow for carrying forward of earlier errors to permit marks to be gained for later solutions that are incorrect, but have used the right method with the wrong value entered. This is one of the main strengths of e-Mark, together with its ability to identify common errors and deliver tailored feedback, and its compatibility with using Word forms for appropriately formatted solution and feedback sheets.

1.3 Appearance

e-Mark is simply an Excel workbook with six pre-formatted worksheets in it, before the student solution sheets are read in:



Of these, MASTER and SUMMARY have no input from the user, being completely generated by the program. The others require information prior to running the program.

The program proceeds by pressing each of the buttons 1-5 (though possibly skipping button 3 and/or 4) in order.

The MASTER sheet is the one to which all student solutions are copied to, together with colour-coded feedback and marks, and actual solutions. A separate column appears for each student, in the order in which the datasets are allocated on the STUDENTS sheet.

A list of total marks is printed to the SUMMARY sheet. The DATA sheet contains the datasets for the students – this would be copied from the Datagen program if that is used (see Chapter 2).

After reading in all student solution sheets, and marking, the MASTER sheet will look something like the following (student names and numbers are fictional)

H14 = You have forgotten to multiply by 2 - partial marks given. The correct answer is ... 96.

All content on this page from row 5 onwards is automatically added by the macros - before the macro is run this will be a blank page from row 5 onwards.

Do not delete column A

1) Read in ALL student solution sheets (text files) | 1) Read in ALL student solution sheets (Excel files) | 2) Mark all student solution sheets | 3) Amend marks/feedback after making manual changes here | 4) Create mail-merge data table (Word feedback only) | 5) Create summary mark sheet

	Question	Marks	Tolerance	What?	17000001 Jane Able		18000001 Joe Bloggs	18000009 John Brown	17000006 Jill Cash
4	Do not delete this row								
5	1.a	3		Solution	5040		720	120	362880
6				Student Sol.	5040		720	120	362880
7				Tick/cross	✓		✓	✓	✓
8				Mark	3		3	3	3
9				Feedback					
10	1.b	5		Solution	480		96	24	8640
11				Student Sol.	480		48	24	1452
12				Tick/cross	✓		×	✓	×
13				Mark	5		3	5	0
14				Feedback	You have forgotten to multiply by 2 - partial marks given. The correct answer is ... 8640.				
15	1.c	5		Solution	720		144	48	17280
16				Student Sol.	720		144	14	17280
17				Tick/cross	✓		✓	×	✓
18				Mark	5		5	0	5
19				Feedback			The correct answer is ... 48.		
20		Q TOTAL			13		11	8	8

form10 / form1 / MASTER / ERROR / SOLUTIONS / DATA / SUMMARY / STUDENTS

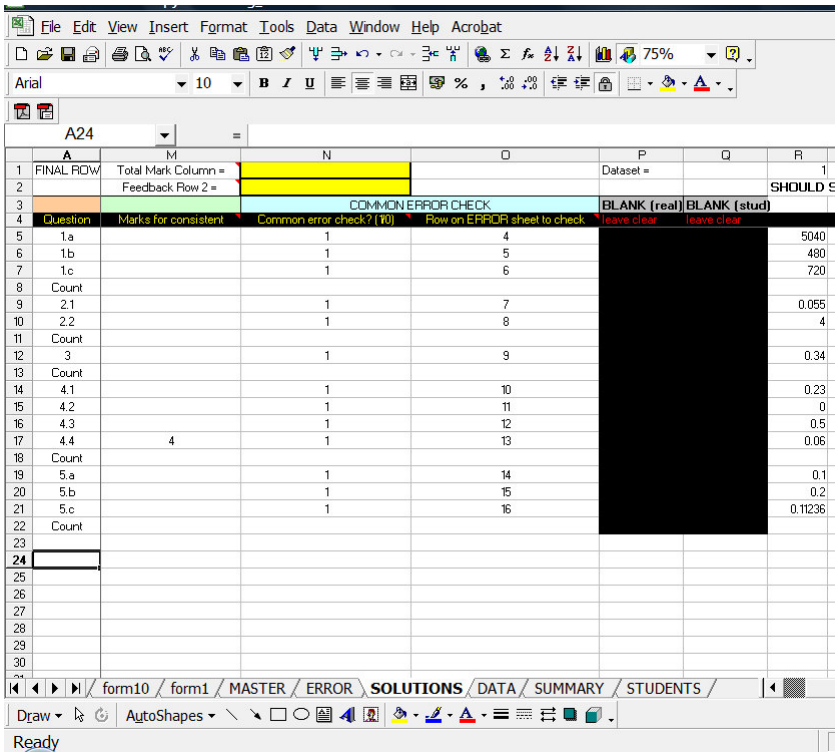
The SOLUTIONS sheet contains all actual solutions for each dataset, together with specified tolerance, and location of solutions on the student's sheet:

K17 = =1-DATA!C12-DATA!C13-DATA!C14+Q14+Q15+Q16

	A	B	C	D	E	F	G	H	I	J	K	L
1	FINAL ROW	TO READ FROM IS		22		Alternative decimal place acceptability is				(integer)	Total Mark Row =	
2											Feedback Row 1 =	
3		GENERAL		LOCATION ON STUDENT SHEET		TICK/CROSS PROVISION		FEEDBACK PROVISION		RULES FOR CONSISTENT ERROR MARKING		
4	Question	Marks	Tolerance	Solution Row	Solution Column	Tick row	Tick column	Feedback row	Feedback column	Consistent error? (1/0)	Consistent calculation	Force consistent?(1/0)
5	1.a	3	0	1	3	-1	3	-1	3	0		
6	1.b	5	0	1	4	-1	4	-1	4	0		
7	1.c	5	0	1	5	-1	5	-1	5	0		
8	Count			0	0	-1		-1				
9	2.1	4	0.001	1	6	-1	6	-1	6	0		
10	2.2	6	0	1	7	-1	7	-1	7	0		
11	Count			0	0	-1		-1				
12	3	6	0	1	8	-1	8	-1	8	0		
13	Count			0	0	-1		-1				
14	4.1	2	0	1	9	-1	9	-1	9	0		
15	4.2	2	0	1	10	-1	10	-1	10	0		
16	4.3	2	0	1	11	-1	11	-1	11	0		
17	4.4	4	0	1	12	-1	12	-1	12	1	0.160	1
18	Count			0	0	-1		-1				
19	5.a	3	0	1	13	-1	13	-1	13	0		
20	5.b	4	0	1	14	-1	14	-1	14			
21	5.c	4	0.001	1	15	-1	15	-1	15			
22	Count			0	0	-1		-1				

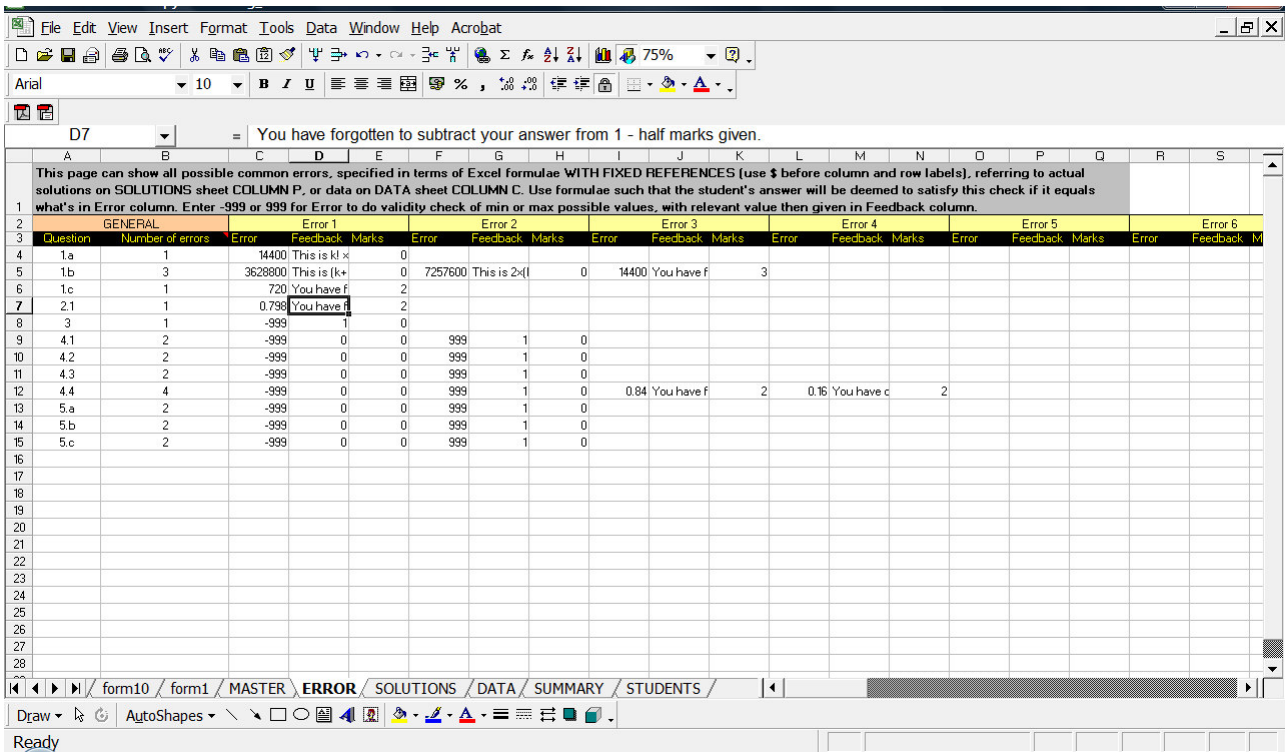
form10 / form1 / MASTER / ERROR / SOLUTIONS / DATA / SUMMARY / STUDENTS

Scrolling along to the right reveals the common error check information (referring to the ERROR sheet), as well as the actual solutions (one column per dataset):



	A	M	N	O	P	Q	R
1	FINAL ROW	Total Mark Column =			Dataset =		1
2		Feedback Row 2 =					SHOULD S
3			COMMON ERROR CHECK		BLANK (real)	BLANK (stud)	
4	Question	Marks for consistent	Common error check? (10)	Row on ERROR sheet to check	leave clear	leave clear	
5	1.a		1	4			5040
6	1.b		1	5			480
7	1.c		1	6			720
8	Count						
9	2.1		1	7			0.055
10	2.2		1	8			4
11	Count						
12	3		1	9			0.34
13	Count						
14	4.1		1	10			0.23
15	4.2		1	11			0
16	4.3		1	12			0.5
17	4.4	4	1	13			0.06
18	Count						
19	5.a		1	14			0.1
20	5.b		1	15			0.2
21	5.c		1	16			0.11236
22	Count						

The ERROR sheet allows specification of many errors, with feedback and marks. It asks for errors to be entered with fixed references, though this isn't actually necessary (it just allows for easier copy and pasting if similar errors are to be specified multiple times, perhaps referring to some of the same components).



D7 = You have forgotten to subtract your answer from 1 - half marks given.

This page can show all possible common errors, specified in terms of Excel formulae WITH FIXED REFERENCES (use $ before column and row labels), referring to actual solutions on SOLUTIONS sheet COLUMN P, or data on DATA sheet COLUMN C. Use formulae such that the student's answer will be deemed to satisfy this check if it equals what's in Error column. Enter -999 or 999 for Error to do validity check of min or max possible values, with relevant value then given in Feedback column.

	GENERAL		Error 1			Error 2			Error 3			Error 4			Error 5			Error 6		
3	Question	Number of errors	Error	Feedback	Marks	Error	Feedback	Marks	Error	Feedback	Marks	Error	Feedback	Marks	Error	Feedback	Marks	Error	Feedback	M
4	1.a	1	14400	This is k! ×	0															
5	1.b	3	3628800	This is (k+	0	7257600	This is 2×(l	0	14400	You have f	3									
6	1.c	1	720	You have f	2															
7	2.1	1	0.798	You have f	2															
8	3	1	-999	1	0															
9	4.1	2	-999	0	0	999	1	0												
10	4.2	2	-999	0	0	999	1	0												
11	4.3	2	-999	0	0	999	1	0												
12	4.4	4	-999	0	0	999	1	0	0.84	You have f	2	0.16	You have c	2						
13	5.a	2	-999	0	0	999	1	0												
14	5.b	2	-999	0	0	999	1	0												
15	5.c	2	-999	0	0	999	1	0												

1.4 Office 2000, 2003 or 2007

The programs have been written to work with all three versions of Excel. However, there are some differences that need to be recognised. Excel 2007 has a larger spreadsheet, and so files in this format cannot be read into an older version of Excel: this includes an older format file that has been opened in Excel 2007 in compatibility mode. If any student solution sheets are likely to be Excel 2007 version, then it is better to have the main e-Mark file as an Excel 2007 version – this means that it must be a macro-enabled 2007 Excel workbook, and therefore has extension .xlsm. If all files will be Excel 2003 or older, then the e-Mark program can also be 2003 version, with extension .xls (though it can also be Excel 2007 with no problem).

If problems are encountered which may be due to the format, it is probably best to save the e-Mark program as a macro-enabled Excel workbook, and work exclusively with 2007. However, any student solution sheets should be saved as an older version, since some students may not have the latest version on their machines.

The program needs to have macros enabled in order to run. In 2007, on opening the workbook a note will appear above the spreadsheet informing you that there are macros in the workbook – clicking on Options will allow you to accept them in this case (this is safer than changing the settings of Office to always allow macros). In 2003 or earlier, set your Office security setting to Medium in Tools>Macro>Security (and restart program), and then you will be asked at the start whether you want to enable the macros or not (select Yes!)

1.5 Before you start

A few words of warning before you embark on e-marking! Using e-Mark for the first time can be daunting, since there are a lot of things to think about. That is the main thing to remember – you **must** plan carefully and allow plenty of time to consider how to best write your assignment, given that the solutions will be marked by computer (i.e. you will want to see the intermediate calculations for larger quantitative questions, that you might not otherwise have looked at). There are issues of data generation to consider – how will you generate many different datasets in a simple way? Will you want them to give similar conclusions to each other, or are you happy for them to be different in many ways? Using some kind of a base dataset seems sensible, but you'll need to think about how to use that to generate multiple versions that still give you the sort of questions and solutions that you want. This takes time! (a lot of it!)

Also, look carefully at your solutions for your datasets. Although you can generate the correct answers (to many decimal places) in a computing package, students may be working out the answers by hand. Some calculations are **greatly affected by rounding errors** – this point is worth stressing as it can affect the ability of the program to correctly mark wrong answers that are consistent with other errors. This does not only apply to students calculating answers by hand. If they are using a computing package to generate numerical solutions, if they perform an earlier step incorrectly then their later answers will be wrong. However, you may want to give them marks for nevertheless doing the later things correctly, albeit with wrong numbers. But your consistent errors (see Section 3.6) will be calculated on the basis of the student's submitted answers – and they will probably be to a smaller number of decimal places than the actual answers. This rounding, when used in other calculations, may well lead to a value which doesn't match what the student has given for the later calculations, even though their answer *is* fully consistent with the other answers. For example, 23.4999999/0.0466 = 504.292, but 23.5/0.047 = 500. Such problems may have to be partially dealt with by specifiying appropriate common errors to pick up (Section 3.5), and doing some manual adjustment post e-marking (Section 4.4).

As already noted above, in Section 1.1, you also need to think very carefully about how you want to approach the marking. Chapter 3 covers how to set up e-Mark, to include things such as consistent and common error checking. But you will need to have decided in advance which questions will depend on others, and how they do so, so that you can determine the formula for entering into the program. You will also need to determine in advance which errors the students may make. And how many marks should they get for a consistent error, or one of the common errors? Many of these aspects are normally considered at the time of physically marking, in response to the set of solutions submitted by a student. Here, they need to be determined, as far as possible, in advance, so they can be coded into the program. Again, this takes time! (a lot of it!)

Some words of advice are given in Chapter 6, but overall the take-home message is that you need to invest a lot of set-up time in e-marking, as least initially, to save you time later on. It's not something that can be easily done at the last minute.

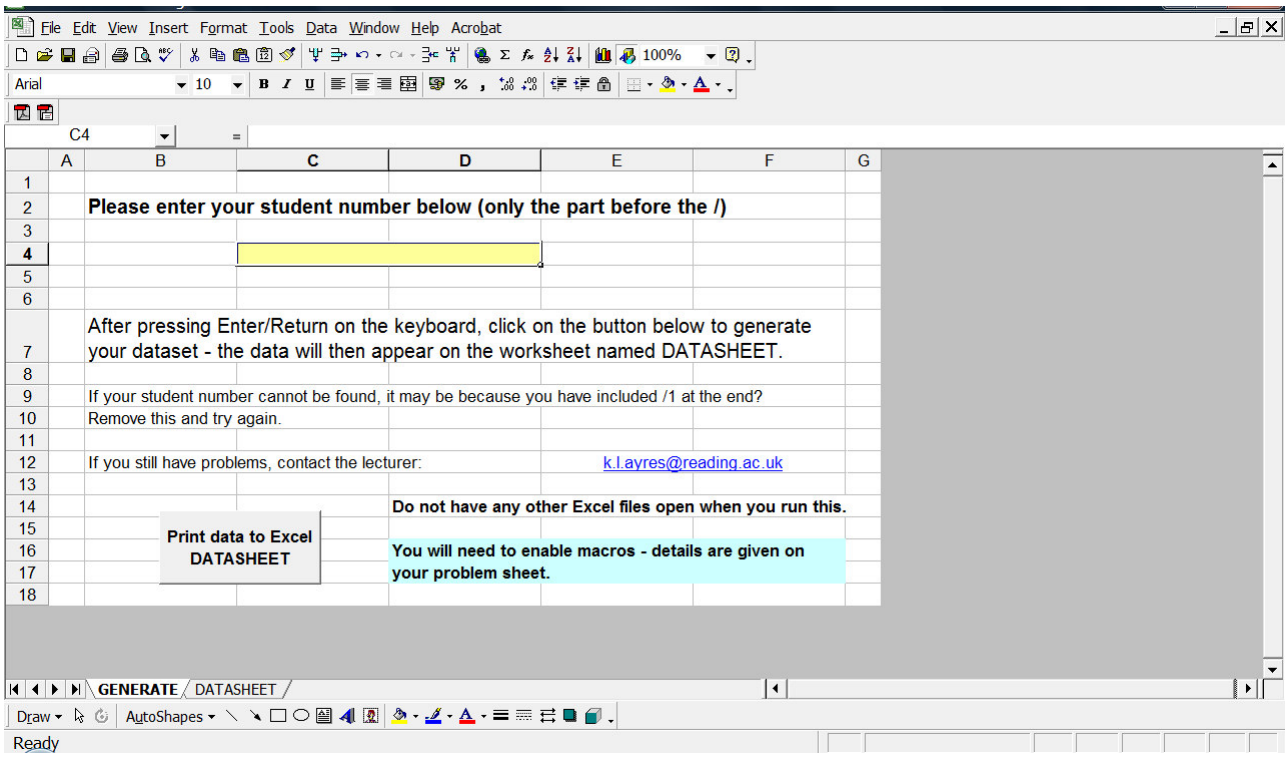
1.6 Overview

This document provides details about how to set-up and use the e-Mark program, plus its accompanying data provision program Datagen, and Word form processor Master.doc. Chapter 2 focuses on the Datagen program. Chapters 3 and 4 cover setting up e-Mark, and running it, respectively. Chapter 5 covers the use of Word forms for student solution sheets and feedback. Chapter 6 provides some general tips about use, and Chapter 7 briefly mentions areas for possible improvement. All program code is included in the Appendices.

2. Dataset provision

One of the important components of the auto-marking program described here is the ability to have individualised solutions per student, i.e. each student can have different data for an assignment, or (within certain constraints) could have different questions.

The problem remains as to how to quickly deliver an individualised data sheet to a student: with a class of 50 or more, e-mailing each sheet to a student, or even asking them to physically collect one that has been printed out, quickly becomes too time-consuming. To solve this problem, the Excel program Datagen has been put together – this is an Excel workbook with macros. The front screen is as follows:



The student enters their ID number in the yellow box, and clicks on the action button. Their own dataset is then retrieved from a hidden worksheet, and printed in whatever format the lecturer wishes, choosing from:

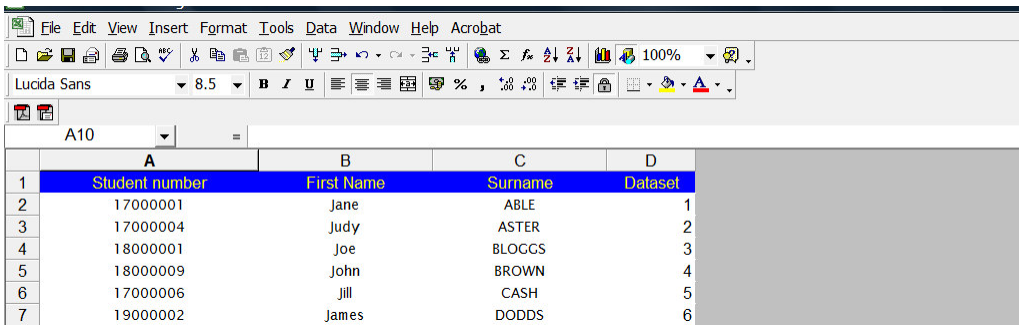
- Pre-formatted Excel worksheet called DATASHEET (as in the example above)
- New Excel worksheet (with variable names at top of each column)
- Text file (with variable names at top of each column)
- Comma-delimited (CSV) file (with variable names at top of each column)

To customise this front page, first unprotect the Datagen workbook, and also the GENERATE sheet. You should add your own contact details, and possibly add other information to the sheet. You will need to link in the relevant macro for the choice of dataset delivery you want and rename the button and information in row 7 (see below, Section 2.2).

2.1 Entering datasets

Datasets are specified by "question", in other words a distinct set of data. This may truly be a question that requires a dataset with many values and many variables. Else it could be a part of a question, and it could only consist of a single value. The nature of the datasets to be made available will depend on the assignment itself. Different types of datasets/datasheets can be accounted for here, and they are specified in the same way in the program. Note that there is a limit of 1000 "questions", i.e. datasets that can be generated for each student. In terms of the number of possible datasets that can be loaded into Datagen per question (i.e. the number of students who can each have an individualised datasheet), this is about 240 for an Excel 2003 file, and many more for Excel 2007 – it is limited by the number of available columns on the worksheet to the right of column G.

To enter the datasets and student details, first unprotect the Datagen workbook, if it is not already unprotected from above. Unhide the sheets DATA and STUDENTS. On the STUDENTS sheet, add details of your classlist, e.g.



	A	B	C	D
1	Student number	First Name	Surname	Dataset
2	17000001	Jane	ABLE	1
3	17000004	Judy	ASTER	2
4	18000001	Joe	BLOGGS	3
5	18000009	John	BROWN	4
6	17000006	Jill	CASH	5
7	19000002	James	DODDS	6

Row 1 is already specified in the program, so you just need to fill in row 2 onwards. Datasets should be numbered from 1 onwards. These numbers refer directly to the columns on the DATA sheet.

Once this sheet has been completed, move to the DATA sheet. You need to have your many datasets worked out already – this program does not generate them, only makes the relevant one available to each student!

The DATA sheet is set out as follows:

Question — Enter a label for the question that requires data. It does not matter what this label is, only that something is entered for each distinct set of data/values that are to be generated, since the program detects non-empty cells.

For FILE
Filename — Needed only for outputting to text or CSV file – the name of the file to save the data to (file will be created at the time the macro runs – if it exists already, the program may fail). Ignored otherwise.

For FILE or new Excel sheet
Columns of data
Specifies the number of columns of data to print to file or Excel sheet – not needed for printing to pre-formatted DATASHEET. This allows for multiple variables/columns of data to be present in a dataset.

Variable Name

Specifies the name of the variable to be printed at the top of the column of data, for file or new Excel sheet output. Not used when data are printed to the pre-formatted DATASHEET, but may nevertheless be useful to enter something here, to remind what the values relate to (it will be ignored in the program in this case).

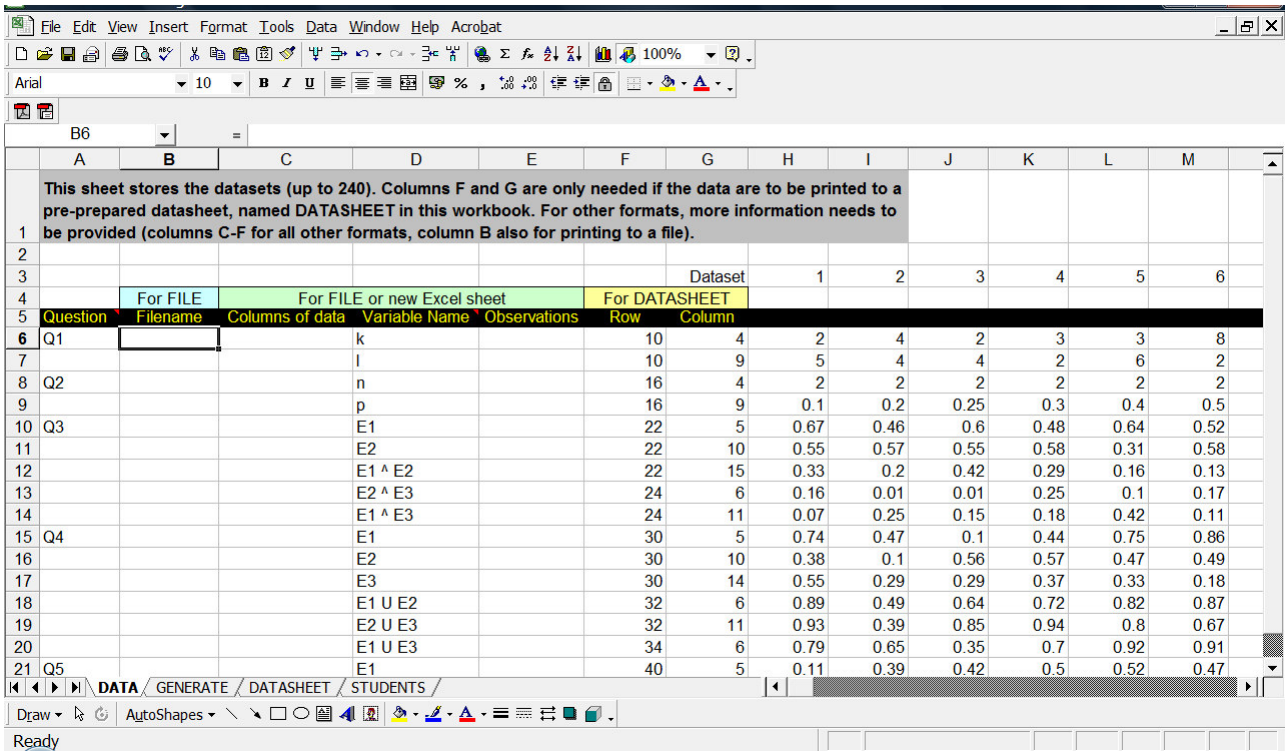
Observations Specifies the number of observations in the dataset – the program knows then how many values to read in from the specified dataset before moving on to the next variable. Not needed when printing to pre-formatted DATASHEET.

For DATASHEET

Row Row number for this data observation to be printed onto the pre-formatted DATASHEET. Not needed for other output types.

Column Column number for this data observation to be printed onto the pre-formatted DATASHEET. Not needed for other output types. These two values together allow the relevant value to be placed in the correct place on DATASHEET.

An example relevant to the case when printing to a pre-formatted datasheet (see Section 2.2) is given below.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	This sheet stores the datasets (up to 240). Columns F and G are only needed if the data are to be printed to a pre-prepared datasheet, named DATASHEET in this workbook. For other formats, more information needs to be provided (columns C-F for all other formats, column B also for printing to a file).												
2													
3							Dataset	1	2	3	4	5	6
4		For FILE	For FILE or new Excel sheet			For DATASHEET							
5	Question	Filename	Columns of data	Variable Name	Observations	Row	Column						
6	Q1			k		10	4	2	4	2	3	3	8
7				l		10	9	5	4	4	2	6	2
8	Q2			n		16	4	2	2	2	2	2	2
9				p		16	9	0.1	0.2	0.25	0.3	0.4	0.5
10	Q3			E1		22	5	0.67	0.46	0.6	0.48	0.64	0.52
11				E2		22	10	0.55	0.57	0.55	0.58	0.31	0.58
12				E1 ^ E2		22	15	0.33	0.2	0.42	0.29	0.16	0.13
13				E2 ^ E3		24	6	0.16	0.01	0.01	0.25	0.1	0.17
14				E1 ^ E3		24	11	0.07	0.25	0.15	0.18	0.42	0.11
15	Q4			E1		30	5	0.74	0.47	0.1	0.44	0.75	0.86
16				E2		30	10	0.38	0.1	0.56	0.57	0.47	0.49
17				E3		30	14	0.55	0.29	0.29	0.37	0.33	0.18
18				E1 U E2		32	6	0.89	0.49	0.64	0.72	0.82	0.87
19				E2 U E3		32	11	0.93	0.39	0.85	0.94	0.8	0.67
20				E1 U E3		34	6	0.79	0.65	0.35	0.7	0.92	0.91
21	Q5			E1		40	5	0.11	0.39	0.42	0.5	0.52	0.47

Columns H onwards specify the actual datasets. Each dataset occupies a single column only. Its entries relate to the variables and number of observations specified. In the above example, it is easy to see how each value in each dataset relates to the questions and variable names/labels for data.

An example relevant to the case when printing to a text file (see Section 2.2) is given below. Here there is only one dataset, but there are four variables, each with 190 observations. The first variable is named Year, and the values are identical for each dataset (hence are the same in each column H onwards).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	This sheet stores the datasets (up to 240). Columns F and G are only needed if the data are to be printed to a pre-prepared datasheet, named DATASHEET in this workbook. For other formats, more information needs to be provided (columns C-F for all other formats, column B also for printing to a file).												
2													
3							Dataset	1	2	3	4	5	6
4		For FILE	For FILE or new Excel sheet			For DATASHEET							
5	Question	Filename	Columns of data	Variable Name	Observations	Row	Column						
6	Q1	data1.txt	4	Year	190			1781	1781	1781	1781	1781	1781
7								1782	1782	1782	1782	1782	1782
8								1783	1783	1783	1783	1783	1783
9								1784	1784	1784	1784	1784	1784
10								1785	1785	1785	1785	1785	1785
11								1786	1786	1786	1786	1786	1786
12								1787	1787	1787	1787	1787	1787
13								1788	1788	1788	1788	1788	1788
14								1789	1789	1789	1789	1789	1789
15								1790	1790	1790	1790	1790	1790
16								1791	1791	1791	1791	1791	1791
17								1792	1792	1792	1792	1792	1792
18								1793	1793	1793	1793	1793	1793
19								1794	1794	1794	1794	1794	1794
20								1795	1795	1795	1795	1795	1795
21								1796	1796	1796	1796	1796	1796

Scrolling down the screen reveals that row 196 is where the second variable commences. Here, the new variable name must be stated (CET in this example) and also the number of observations stated (again, 190). Columns A-C are left empty because this is just the second column of a 4-column dataset, so filename etc have not changed. Only if we needed to specify data for the second question would we add something to columns A-C again.

It is worth stating again that for file and new Excel sheet delivery methods, the variable names need to be stated for each variable on the row where that variable's data begins, together with the number of observations that are in the dataset for that variable (i.e. the number of rows of data in the dataset's column that relate to that variable, before the next variable's data begins).

	A	B	C	D	E	F	G	H	I	J	K	L	M
193								1968	1968	1968	1968	1968	1968
194								1969	1969	1969	1969	1969	1969
195								1970	1970	1970	1970	1970	1970
196				CET	190			10.2	10.62	10.8	10.49	9.65	10.74
197								8.01	8.07	7.85	7.84	8.08	7.55
198								9.28	9.31	9.83	9.47	9.25	9.42
199								7.83	7.56	8.05	7.73	7.92	8.54
200								8.54	8.67	8.93	8.68	8	8.49
201								8.25	8.59	8.07	8.98	7.89	8.04
202								9.28	9.27	9.26	8.73	9.62	9.11
203								9.21	9.02	9.12	9.27	8.83	9.33

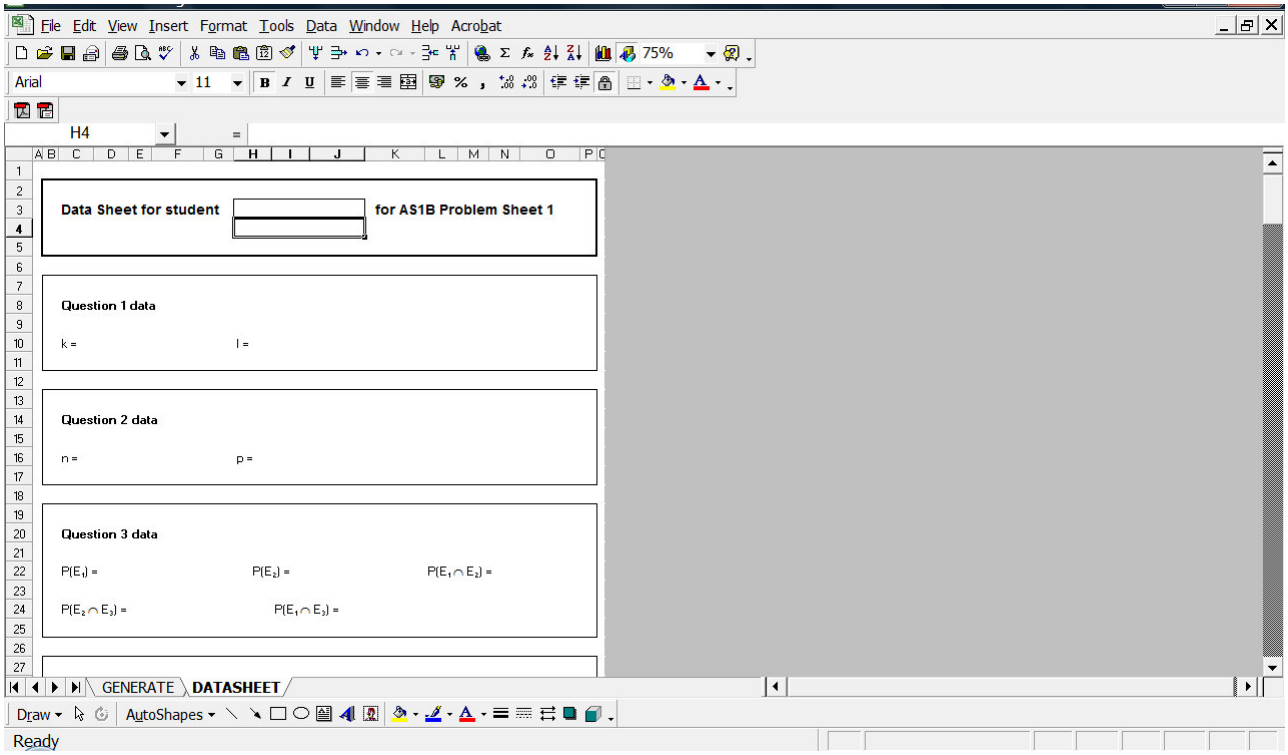
Continuing with this example, row 386 has the next variable name specified, together with 190, and row 576 has the final variable name specified, together with 190.

The program knows to cease producing a data file when:

- For pre-formatted DATASHEETs – it reaches an empty cell in column F
- For other methods – it reaches the final value for the number of observations and variables specified in columns D and E, and the next cell in column A is empty. Therefore each question (distinct dataset to be generated) must have something entered into column A.

2.2 Output format

For assignments with many questions that each have only a small number of values that must be individualised, the best approach is probably to output to a pre-formatted Excel worksheet. This should be named DATASHEET and can be formatted in any way desired, except that (if the Visual Basic code is not going to be changed) a box for the student's ID number should be provided with cell reference H3, and a box for their name with cell reference H4: these can involve merged cells, such as shown below, if formatting is improved in this way.



The code for the four macros that output in different formats is given in Appendix A. These are PrintDataSheet, PrintSheet, PrintText, PrintCSV. All four are already present in the Datagen program, and only require the relevant one to be linked in.

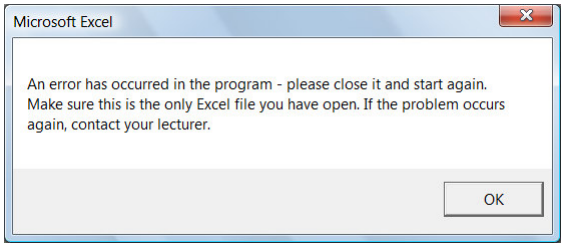
Depending on the method you want to use, you should right-click on the action button on the GENERATE worksheet, and assign the relevant macro to that button. Right-clicking on the button followed by left-clicking on the text will also allow you to edit the text on it, relevant to the action that the button will take, e.g. "Print data to Excel DATASHEET", "Print data to new Excel worksheet", "Print data to tab-delimited text file", "Print data to CSV file" (or text of your own choosing). Information can be given in row 7 of the GENERATE sheet regarding the data delivery. In the example above, it informs the student that the data will appear on the DATASHEET worksheet – if you are using a different macro, then add some relevant text in cell B7 to make it clear that the data will e.g. be printed to a file.

If printing to a pre-formatted sheet is not to be used, then the DATASHEET sheet should be deleted. Sheets DATA and STUDENTS must remain, and will be hidden on the next run of the macro (so take care not to accidentally left-click on the macro action button when you're trying to update it, since the code will run and re-protect and hide the sheets!) This is the quickest way to re-hide the two sheets – click on the action button while leaving the student ID box blank.

For printing to a new Excel sheet, or a file, a new sheet or file will be used for each distinct "question" as denoted on the DATA sheet (this may truly be a new question, or part of a question,

depending on where a new dataset is required). The variable names, as specified on the DATA sheet, will be given at the top of each column. For files, these will be generated in the same folder that the Datagen program is residing in (hence students will need to download this file first and save it in their directory, else the relevant path may not be valid). The path is printed in a message box once the data has been generated.

If the program fails for some reason, an error message appears, asking the student to either remove any previously created files and try again, or contact the lecturer, e.g.



This is produced as a result of the error trapping code placed in the program (On Error Goto), and it prevents students inadvertently accessing the code or hidden sheets if the program fails mid execution.

2.3 Protecting the file

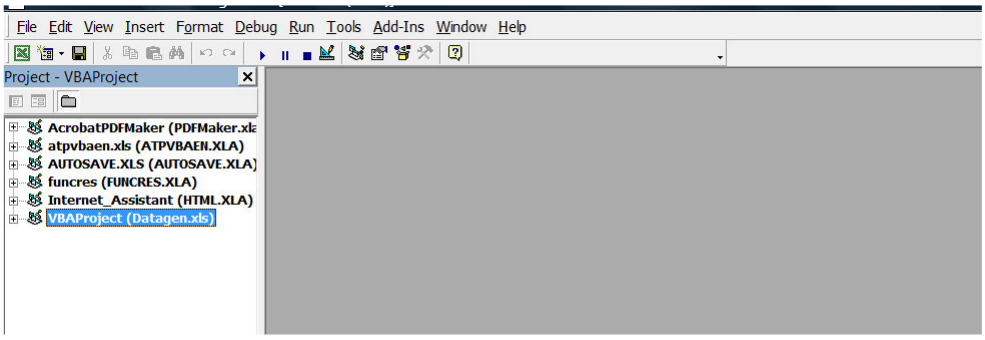
Although it does not matter if students obtain datasets for others, it is nevertheless desirable to try to limit this. The easiest way to try to prevent this is to protect the workbook, and also the code (since the code must include the password for the workbook, as the workbook needs to be unprotected in order to access certain components while the macro is being executed).

Protect the sheet GENERATE and the entire workbook as you would usually – make sure you use the same password that is specified in the Visual Basic code though! Protecting the GENERATE sheet and the workbook prevents students from accidentally deleting the generate button for example. The easiest way to do this is actually just to click on the macro button (as described above), since when the code is run, part of it protects the sheets and workbook anyway. But make a note of the password that you have put in the code first, else you won't be able to unprotect it!

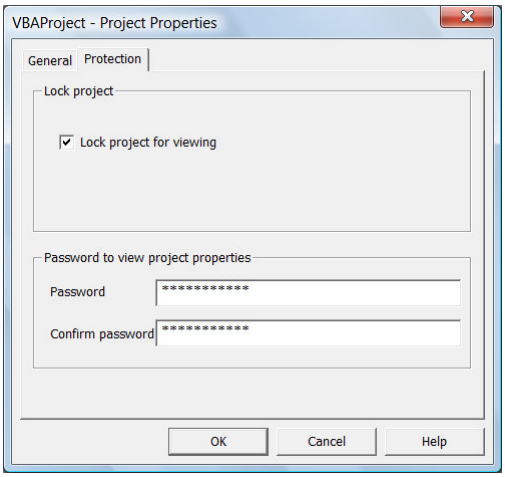
To protect the code, assuming that it has been unprotected at some point, possibly to update it, you should do the following.

Excel 2000, 2003

Go to Tools>Macro>Visual Basic Editor. Click the minus sign next to the VBAProject entry in the Project pane, to close it up (see below).



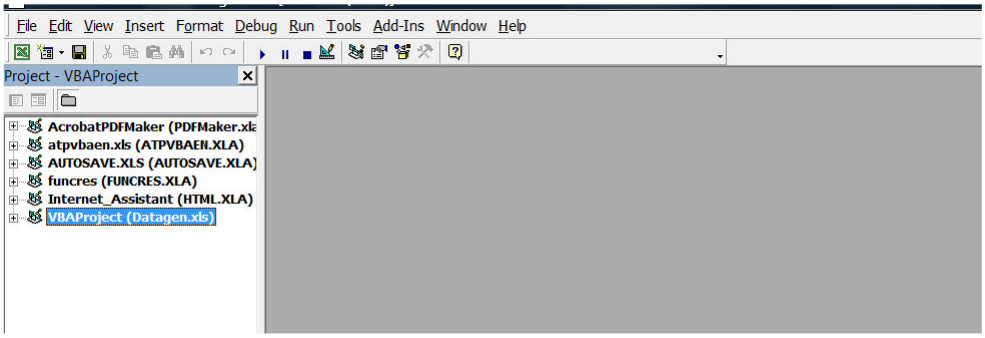
Select Tools>VBA Project – Project Properties followed by the Protection tab. Check Lock Project for Viewing, and enter your password, thus



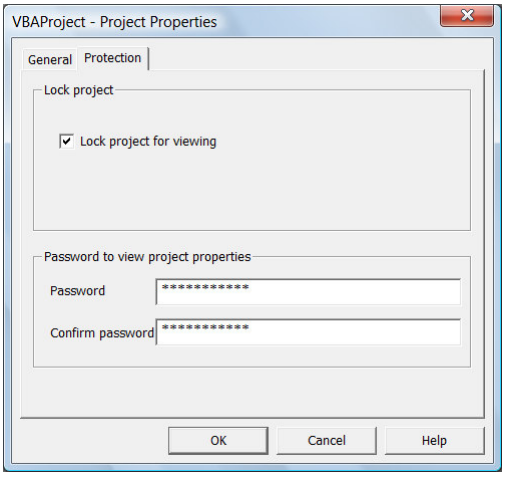
This will have no effect until the workbook is saved and closed, and re-opened. Once it is opened, the code is then password protected, and cannot easily be accessed. It is not, however, likely to be completely unhackable!

Excel 2007

From the Developer ribbon, click on Visual Basic (you may need to enable this facility if it isn't already enabled). Click the minus sign next to the VBAProject entry in the Project pane, to close it up (see below).



Select Tools>VBA Project Properties followed by the Protection tab. Check Lock Project for Viewing, and enter your password, thus



This will have no effect until the workbook is saved and closed, and re-opened. Once it is opened, the code is then password protected, and cannot easily be accessed. It is not, however, likely to be completely unhackable!

2.4 Possible problems

Experience with using this program has shown that some students struggle to get it working as they may not follow the instructions the first time. For example, they may try to run it from within a Virtual Learning Environment (VLE) such as Blackboard, where it is unlikely to work properly. Also, running it multiple times will likely generate an error once files have already been created – they need to be deleted and then the program run again, if it needs to be. Furthermore, it is not designed to be run while other Excel workbooks are open, and so it is likely to fail or behave incorrectly if all other workbooks have not been shut down before the macro button is clicked. It is not clear how the macros will perform on a Macintosh computer – they will probably not work when the file is opened in a package other than Microsoft Excel. In this case, it is probably safer to run it on University machines that are known to have the correct software on them. These issues need to be made clear to students, and preferably all students should try to obtain their own dummy set of data in advance of having to use it to obtain actual data for assignments. They should also obtain their data in good time before the deadline for the assignment!

3. e-Mark – initial set-up

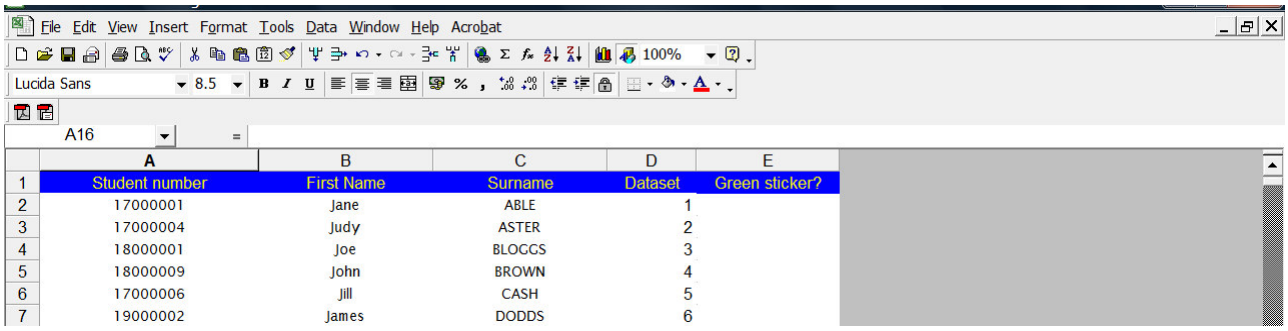
3.1 Introduction

The program e-Mark is a collection of macros written in Visual Basic in Excel – all code at time of writing this document is given in Appendix B.

The program requires setting up for each assignment that it is being used for, including entering of solutions, and specification of errors to check for. This can take a lot of time, and so should be done in advance of the assignment being handed out! The key thing to remember is that a lot of thought is needed when specifying the errors to check for, second-guessing what students may do wrong, and deciding on the strategy for dealing with such errors. This document does not consider these points in detail, but focuses instead on how to set-up the program to implement the strategy decided upon.

3.2 Students

The first part of the program to set-up is the list of students and their respective datasheets on the STUDENTS worksheet. This sheet is identical to that described for the Datagen program in Chapter 2, but it has an additional column included, labelled 'Green sticker?'. This column is a disability flag, indicating whether hand-written work would have been marked according to some degree of flexibility regarding spelling and grammar. For example



	A	B	C	D	E
1	Student number	First Name	Surname	Dataset	Green sticker?
2	17000001	Jane	ABLE	1	
3	17000004	Judy	ASTER	2	
4	18000001	Joe	BLOGGS	3	
5	18000009	John	BROWN	4	
6	17000006	Jill	CASH	5	
7	19000002	James	DODDS	6	

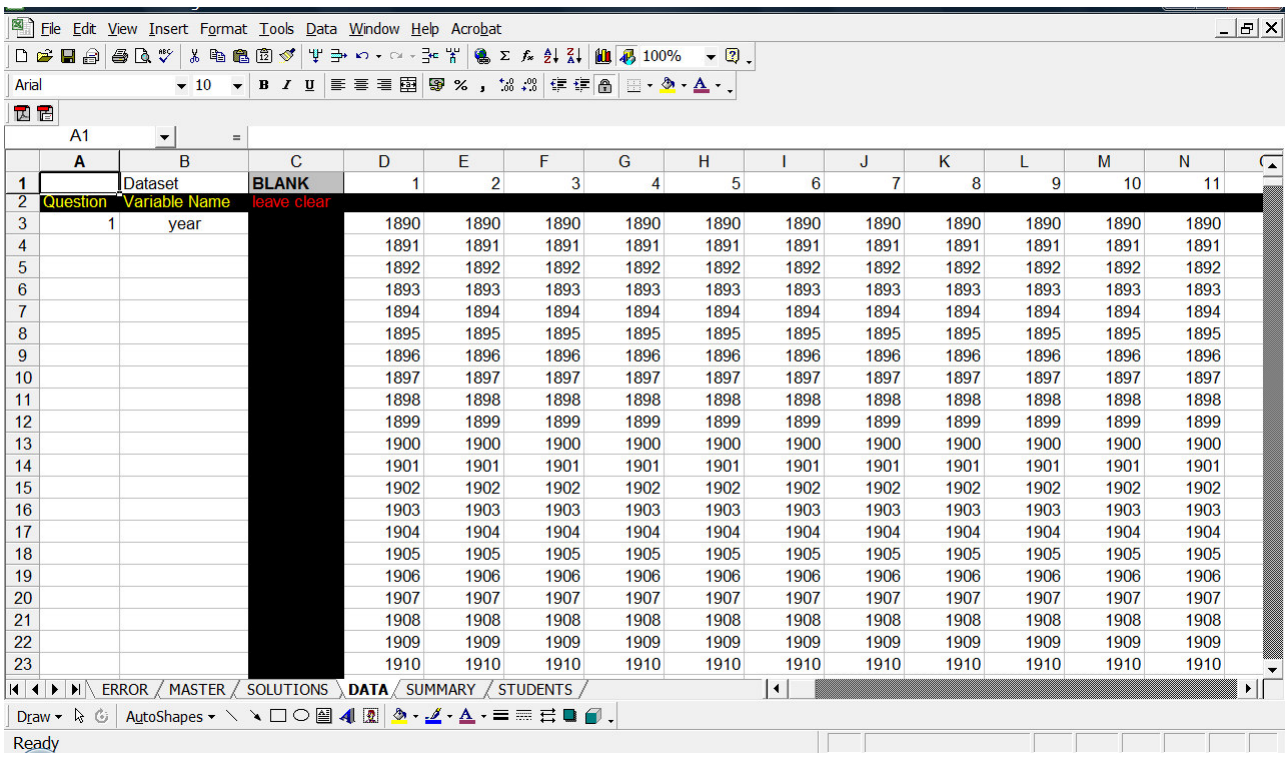
The column should be left blank if no allowance is to be made, or a 1 entered if there should be. Note that the program does not make any allowance itself, but for those students with a flag of 1, a feedback statement will be printed at the end of the assignment that says "Your work has been marked assuming that you have attached a green sticker to it, and appropriate allowance made according to the text on that sticker." If there is an alternative way of identifying such students, then the text will need to be amended in the Visual Basic code, in the MarkSheets macro.

Having the feedback added is useful in two ways – first, it will flag to the marker that appropriate allowance needs to be made when auditing the results of the program, as the feedback is printed to the MASTER sheet (see Section 4.3). It will also inform the student that proper consideration has been made when their work has been marked.

In order to complete the STUDENTS sheet, it is probably easiest if the rows are just copied from the Datagen program, if that has been used, so that there is no accidental discrepancy between names and datasheet numbers. Green sticker entries can then be added afterwards.

3.3 Data

The DATA sheet contains all of the individualised datasheets for the students (possibly more), with one column per datasheet. This is in the same format as in the Datagen program, and so the relevant cells from that sheet can be copied straight into this one (note that the DATA sheet here must have column C blank, as the consistent and common errors will read from this). An example is shown below.



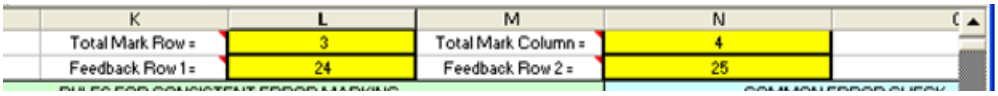
	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		Dataset	BLANK	1	2	3	4	5	6	7	8	9	10	11
2	Question	Variable Name	leave clear											
3	1	year		1890	1890	1890	1890	1890	1890	1890	1890	1890	1890	1890
4				1891	1891	1891	1891	1891	1891	1891	1891	1891	1891	1891
5				1892	1892	1892	1892	1892	1892	1892	1892	1892	1892	1892
6				1893	1893	1893	1893	1893	1893	1893	1893	1893	1893	1893
7				1894	1894	1894	1894	1894	1894	1894	1894	1894	1894	1894
8				1895	1895	1895	1895	1895	1895	1895	1895	1895	1895	1895
9				1896	1896	1896	1896	1896	1896	1896	1896	1896	1896	1896
10				1897	1897	1897	1897	1897	1897	1897	1897	1897	1897	1897
11				1898	1898	1898	1898	1898	1898	1898	1898	1898	1898	1898
12				1899	1899	1899	1899	1899	1899	1899	1899	1899	1899	1899
13				1900	1900	1900	1900	1900	1900	1900	1900	1900	1900	1900
14				1901	1901	1901	1901	1901	1901	1901	1901	1901	1901	1901
15				1902	1902	1902	1902	1902	1902	1902	1902	1902	1902	1902
16				1903	1903	1903	1903	1903	1903	1903	1903	1903	1903	1903
17				1904	1904	1904	1904	1904	1904	1904	1904	1904	1904	1904
18				1905	1905	1905	1905	1905	1905	1905	1905	1905	1905	1905
19				1906	1906	1906	1906	1906	1906	1906	1906	1906	1906	1906
20				1907	1907	1907	1907	1907	1907	1907	1907	1907	1907	1907
21				1908	1908	1908	1908	1908	1908	1908	1908	1908	1908	1908
22				1909	1909	1909	1909	1909	1909	1909	1909	1909	1909	1909
23				1910	1910	1910	1910	1910	1910	1910	1910	1910	1910	1910

Note that columns A and B are not used within e-Mark, but are nevertheless useful to complete, to help locate relevant data entries when specifiying common or consistent errors to check for.

3.4 Student solutions – Word or Excel?

When the program executes it asks whether feedback is to be provided through Word forms or not. If Word forms are not to be used, then it is assumed that pre-formatted Excel sheets are to be used (students would have handed in their solutions in this format, and feedback will be added prior to their return). For Excel sheets, the row locations of the cells where the two general pieces of feedback (see Section 4.4) are to be placed must be given (the column is assumed to be 1), and also the row and column where the final total mark is to be placed. These locations (in the form of the row and column number for the appropriate cell) must be specified at the top of the SOLUTIONS sheet (in the yellow cells). If feedback is to be handed back using Word forms, then these cells are ignored, and do not need to be completed.

For example, if the two feedback statements are to be placed in cells A24 and A25 on the students' sheets, and the overall total mark in D3, we would have



K	L	M	N
Total Mark Row =	3	Total Mark Column =	4
Feedback Row 1 =	24	Feedback Row 2 =	25

3.5 Solutions, marks and tolerance

The main content of e-Mark is specified on the SOLUTIONS sheet, and general information columns are grouped together under the banners of GENERAL, LOCATION ON STUDENT SHEET, TICK/CROSS PROVISION, FEEDBACK PROVISION, RULES FOR CONSISTENT ERROR MARKING and COMMON ERROR CHECK.

The largest component to enter is the set of solutions for every student. These are entered after the black columns (these two columns, P and Q, must be left blank as this is where the student and actual solutions are copied to during the running of the program). There is one column per set of solutions, and the solutions for each column after this point should match the relevant datasheet number. Note that text answers that start with an equals sign, such as =>, cannot be typed directly into Excel as it will view them as error in a formula. If you have answers of this type, then you will need to first type them into a text file, and copy and paste them from there. There will not be problems when the program is run, as e-Mark will treat them as text and handle them correctly.

The form of the SOLUTIONS sheet is such that the first column is reserved for a label for the question pertaining to that row in the worksheet. This label can be anything, since it is not used in the marking process. The next two columns, also coming under the banner of GENERAL, allow for the marks to be entered for that question (these can be decimal, not only integer), as well as the tolerance that will be used for determining whether a student's solution is correct or not. That is, the tolerance e is used in the following

> Let a be the student's solution, and let b be the actual solution. If $\text{abs}(a - b) \leq e$ then the student's solution is correct, where abs() denotes the absolute value of a number.

To avoid problems with Excel marking something wrong when it is directly on the boundary of the tolerance, a small value (0.0000000001) is automatically added to every tolerance within the code. This should be remembered if the actual tolerance set is close to, or even less than, this small value, since the value used may be noticeably larger than the value intended in such cases.

When determining the value for the tolerance, consider that different roundings in intermediate steps may result in different final answers. The tolerance should be set to try to address this issue. The bigger the tolerance is, the less likely it will be to have to fix solutions marked incorrect (due to rounding) by hand later, but the more likely it is that an incorrect answer will be marked correct by the program (provided it falls within the tolerance interval, it will be marked correct, however that answer came about). Bearing this in mind, it is clear then that the program is better suited for quantative answers that are not greatly affected by different possible roundings in their calculation, allowing sensible tolerances to be set.

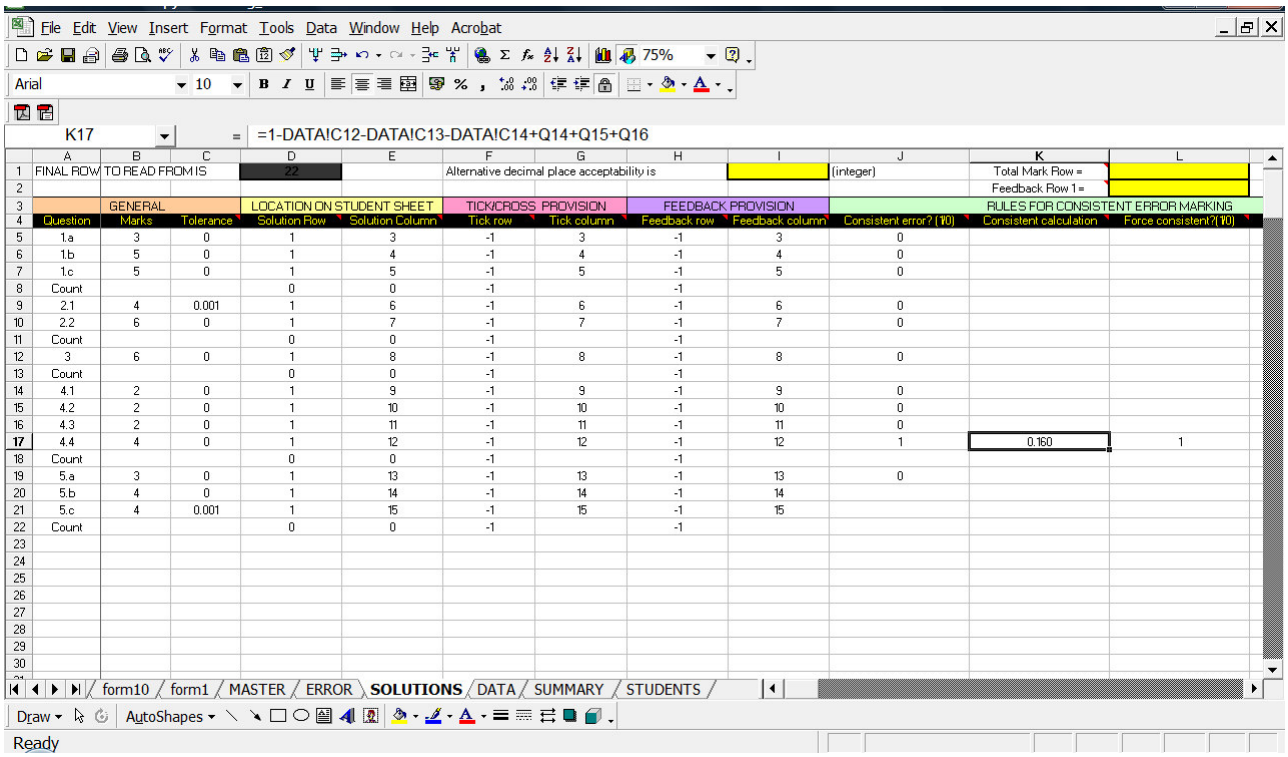
If a tolerance is not wanted, or needed, then the cell in this column for the question can be left blank, or a 0 formally entered.

The LOCATION ON STUDENT SHEET columns specify the references of the locations of the student's solutions for each question, on their uploaded sheets. If their solutions were submitted on pre-formatted Excel sheets, then both a row and column reference number is needed (integer values). If solutions were submitted using Word forms, and the data extracted using the Master Word file (Section 5.3), then all solutions will be on row 1 of the uploaded sheet. Therefore, if Word forms have been used, all values in the Solution Row column will be 1.

An exception to these rules for entering the location references is when something is to be marked outside of e-Mark, and marks added in by hand afterwards. For example, a graph can be submitted

and marked by hand, or a discursive passage. In such cases, to ensure that a blank space is left in which to enter a tick/cross, marks and feedback, a minus 1, –1, should be entered in the Solution Row column. **It is important that, if using Word files for submission of student work, a blank cell (or some similar pre-filled in cell) is included in the student's solution file, so that their solutions match up with those listed in order on the SOLUTIONS sheet).** Furthermore, if intermediate mark totals are to be printed (for example, marks per question in a larger assessment), then appropriate spaces can be flagged here by entering a zero, 0, into the Solution Row column. These 0 entries will form the basis of appropriate formatting in the MASTER sheet when the program is run, allowing the intermediate mark totals to be displayed there. Without 0s entered on this sheet, these intermediate totals cannot be generated. **Note also: the final row on the SOLUTIONS sheet must have a 0 in the Solution Row column.** The total mark for the assignment is calculated as the sum of the intermediate marks, so there must be at least one intermediate mark (including the final one).

The TICK/CROSS PROVISION columns are formatted similarly, in that integers that index the cells where ticks and crosses will be placed on the student sheets are specified here. If Word forms are to be used when returning student work, then the Tick Row column can be left blank, or can contain any numbers, since this column will not be used (for example, a –1 could be entered as a reminder that Word forms are to be used, as shown in the screenshot below).



	A	B	C	D	E	F	G	H	I	J	K	L
1	FINAL ROW TO READ FROM IS			22		Alternative decimal place acceptability is				(integer)	Total Mark Row =	
2											Feedback Row 1 =	
3		GENERAL		LOCATION ON STUDENT SHEET		TICK/CROSS PROVISION		FEEDBACK PROVISION		RULES FOR CONSISTENT ERROR MARKING		
4	Question	Marks	Tolerance	Solution Row	Solution Column	Tick row	Tick column	Feedback row	Feedback column	Consistent error? (Y0)	Consistent calculation	Force consistent?(Y0)
5	1.a	3	0	1	3	-1	3	-1	3	0		
6	1.b	5	0	1	4	-1	4	-1	4	0		
7	1.c	5	0	1	5	-1	5	-1	5	0		
8	Count			0	0	-1		-1				
9	2.1	4	0.001	1	6	-1	6	-1	6	0		
10	2.2	6	0	1	7	-1	7	-1	7	0		
11	Count			0	0	-1		-1				
12	3	6	0	1	8	-1	8	-1	8	0		
13	Count			0	0	-1		-1				
14	4.1	2	0	1	9	-1	9	-1	9	0		
15	4.2	2	0	1	10	-1	10	-1	10	0		
16	4.3	2	0	1	11	-1	11	-1	11	0		
17	4.4	4	0	1	12	-1	12	-1	12	1	0.160	1
18	Count			0	0	-1		-1				
19	5.a	3	0	1	13	-1	13	-1	13	0		
20	5.b	4	0	1	14	-1	14	-1	14			
21	5.c	4	0.001	1	15	-1	15	-1	15			
22	Count			0	0	-1		-1				

The FEEDBACK PROVISION columns are again formatted in a similar way – these index the locations that the feedback for each question will be printed to on the student's solution sheet. Again, where Word forms are to be used, the Feedback Row column can be left blank, or anything entered, as it will not be used.

Note that the intermediate total mark rows (8, 11, 13 and 22 in the above screenshot) do not need an entry in FEEDBACK PROVISION columns (–1 is only shown above as a reminder that Word forms are to be used). However, if Word forms are not used, then the location of the intermediate total on the student sheet is specified by the values in the TICK/CROSS PROVISION columns, and

so these **must** be filled in if Excel sheets are used for providing marked work/feedback to students (but not if Word forms are used).

One of the facilities in the program is to allow for a check of a student's incorrect solution against what the answer should be if it were rounded to a given number of decimal places. For example, if the answer is 0.346 but the student has entered their solution as 0.35, this would be marked wrong if the tolerance was set to be less than 0.004. However, the marker can specify that all answers be checked for being correct to two decimal places. If this is the case, then the program will first check that the student's solution is indeed a 2-decimal place value, and then it will check whether it matches the equivalent decimal place rounded version of the actual solution. If it does, the answer is marked correct.

To request that an alternative decimal place be checked during the marking, enter the relevant integer for the number of decimal places in the relevant yellow cell at the top of the SOLUTIONS screen. Non-integer values, non-numeric and blank entries in this cell are all treated as meaning that no alternative decimal place should be performed. Therefore, if it is not desired, leave the cell blank.

Finally, note that cell D1 is not set by the user: this cell contains the final row to read from, as used by e-Mark when it runs, and should not be changed. If it is accidently overwritten, then the formula
=COUNT(E5:E5004)+4
should be re-entered into it. Note that this allows for 5000 questions/solutions/marks to be listed on the SOLUTIONS sheet, and no more.

3.6 Consistent errors

One of the main advantages of e-Mark is its ability to check solutions for their consistency with earlier errors, allowing marks to be awarded for method, and avoiding double punishment for an earlier mistake. However, careful thought needs to go into the specification of these consistent errors, writing down which answers are suitable for having this check, and which other answers they depend on (and how). Note that a consistent error need not just draw upon earlier answers: they can be functions of any answers, and can depend on student solutions, actual answers, and functions of the data, depending on what is appropriate for the question. However, usually they would depend only on student solutions and data.

The consistent error check, one per question, is added to the program in the columns headed RULES FOR CONSISTENT ERROR MARKING. The first of these columns is a flag to indicate whether a consistent error check is to be performed for the question or not: if it is, then a 1 should be entered into the column. If not, then a 0 can be entered, or the cell left blank (as in the earlier screenshot). The third column allows the consistent error check to be forced, that is, the check is performed even if the student has entered the correct solution for that question. In this way, guesses can be identified, and those answers not consistent with earlier answers can be marked incorrect. Note that forcing consistent errors works best when the solutions are qualitative, since the number of decimal places used in the student solutions can affect numerical answers. That is, the consistent error calculation based on fewer decimal places may differ from the actual solution – if the student has the correct solution, they may not match with the consistent error one for this reason. Finally, the marks to be awarded for a consistent error are specified (note that no check is performed to make sure these are not greater than the number per question!) These can be integer or decimal values.

The consistent error itself is entered in the next column, but is only considered if a 1 has been entered earlier. Appropriate use needs to be made of Excel's many functions for calculating the relevant answer to appear here, which makes use of the student's solutions that would have been copied into column Q when the program is run. For example, if a question asks for the average of a set of numbers, and row 5 contains the total of 6 numbers, then the consistent error formula that needs to be entered into column K is =Q5/6 or =$Q$5/6 (note that it is usually preferable to use fixed cell references, with $ signs, to make things easier when cutting and pasting similar formulae into different cells, particularly if they need to refer to some of the same cells as before). Now, whatever value the student has entered for the previous question about total value of the six values, provided their answer for the average is equal to that number divided by 6, they will satisfy the consistent error check (and be awarded the relevant number of marks).

Consistent error checks can be as complicated as the functions available in Excel. For example, in one implementation of the program the lower confidence interval bound for predicting the response for a value of x=8.6 from a regression model is given by
=$Q$35-TINV(0.05,$Q$12)*SQRT($Q$14*((1/38)+((8.6-
AVERAGE(DATA!$C$79:$C$116))^2)*(1/(SUMSQ(DATA!$C$79:$C$116)-
(SUM(DATA!$C$79:$C$116)^2)/38))))
This includes use of the TINV function to obtain the critical value from the t distribution, SUMSQ for obtaining the sum of squares, SUM for sum, and AVERAGE for average, as well as ^2 for squaring a value. It also makes use of the actual data for the student. Note that when the program runs, the student's data is copied into column C of the DATA sheet. Therefore, consistent (and common) error checks can apply calculations to the student's data, by referring to column C of the DATA sheet. To do this, code such as DATA!$C$10 is needed (this being cell C10 on the DATA sheet). Again, fixed cell references are not needed, but make things easier when copying formulae to other cells.

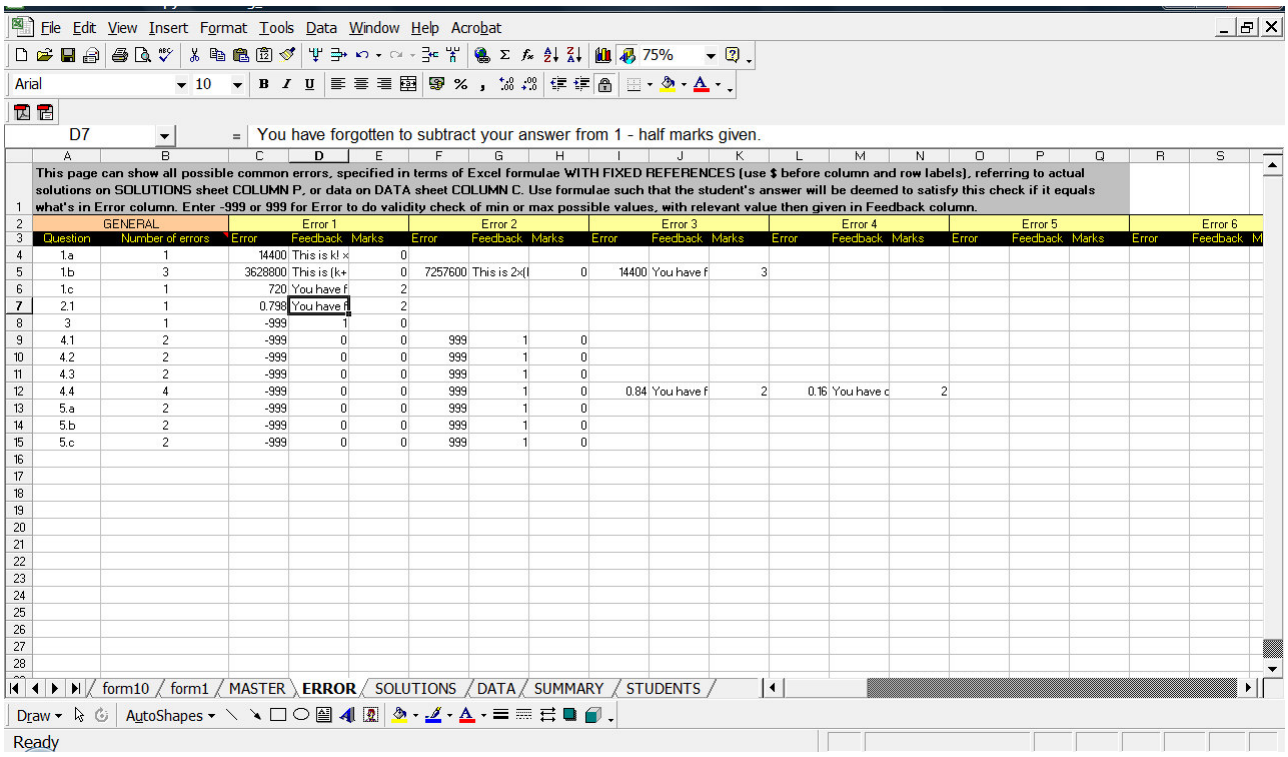
For qualitative answers, use can be made of the IF function, for example the following code contains nested IF statements, to determine whether the answer should be strong, some, weak or no
=IF($Q$32<0.01,"strong",IF($Q$32<0.05,"some",IF($Q$32<0.1,"weak","no")))

To re-iterate, any Excel worksheet functions can be used when specifying consistent errors. Using inbuilt functions can often shorten the formula that needs to be entered (though always check that the same underlying formula is being used by Excel as is being used by the software that generated the actual solutions!) A good safeguard when setting up the SOLUTIONS sheet is to enter the real answers into column Q, plus the relevant dataset into column C of DATA, and then check the consistent error calculations with column Q on the SOLUTIONS sheet – columns K and Q should be identical, allowing for rounding.

Finally, rounding is actually a potentially important issue. Although very complex formulae can be entered for consistent error checks, it should be noted that the ability of e-Mark to catch consistent errors is limited by the need for the student's solution to be within the stated tolerance, which may not be the case if answers vary wildly due to different roundings. Also, there is no alternative decimal place check for consistent errors. In fact, there is actually no check performed to ensure that the other solutions are indeed incorrect! While this should not affect the performance of the program in general, it does cause problems when forcing a consistent error check sometimes, due to loss of precision because of rounding. A change is planned to address this issue.

3.7 Common errors and feedback

A number of checks can be made for errors that we may expect a student to make when completing the question. The intention here is to try to identify reasons why a student's solution is incorrect, so that appropriate feedback can be given, together with marks. A common error check does not have to be added to the program for each question: if one or more is needed for any question, then the appropriate flag (a value of 1) should be added to the column headed Common Error Check? If no check is to be performed, then a 0 can be entered, or the cell left blank. If a 1 is entered, then the program will check the second column to determine which row to read from on the ERROR sheet. It is this row that specifies the actual common errors to check for. Note that row 4 is the first row that can be used, since the ERROR sheet uses the first three rows for information and headings. Recall the following screenshot of the ERROR sheet, shown earlier



This sheet can be completed in the following way. The first column, headed Question, is only a label for the question. The program does not use this column, and it is only included for information purposes. Therefore it does not matter what is entered here.

The second column specifies the number of errors to check for, i.e. the number of column blocks to cycle through. There is no limit to the number of errors that can be checked per question, though only ten blocks have a formatting heading on the ERROR sheet.

There are essentially two different types of common errors to check for. The first is a validity check. Often there will be natural limits for quantitative answers: either a lower limit or an upper limit, or both. These can be specified here quite simply. Under the Error column, enter –999 to denote a lower limit check, or 999 for an upper limit check. In the Feedback column, enter the value for the limit itself (a lower limit of 0 and upper limit of 1 are examples shown in the screenshot above). Default feedback will be used to explain the error that has been made, if the student's solution falls outside these limits.

For common errors, these can be specified in a similar way to consistent errors, making use of Excel's inbuilt functions, and being as complex as required. Again, they can refer to the student's

solutions in column Q (SOLUTIONS!$Q$10 for cell Q10), the actual solutions in column P (SOLUTIONS!$P$10 for cell P10), or the data (on the DATA sheet). Note that SOLUTIONS! is used now in the formulae, as the errors are not specified on the SOLUTIONS sheet but on a different sheet, the ERROR one. The relevant formula needs to be entered into the Error column. Here it is important to note that the student's solution is deemed to satisfy this error if it equals the value that appears in this column. For example, if the actual answer is 0.16 for s^2 but the student enters 0.4, and we expect a common error to be that the student calculates s but forgets to square it, then in the Error column we would enter =SQRT(SOLUTIONS!$P$10) where P10 on the SOLUTIONS sheet contains the actual answer. This cell will then contain the value 0.4, which when compared to the student's answer of 0.4, is "correct". We have therefore identified the error made: the student forgot to square their value of s. In the column headed Feedback, we would enter something appropriate for when this error is identified, e.g. "You have forgotten to square s." Note that we cannot apply any formatting here (e.g. italics), since the program will pick up only the text.

The flexibility with being able to refer to either the student's solutions (column Q) or the actual solutions (column P) allows us to perform various checks. In fact, common error checks could also refer to the consistent error column on the SOLUTIONS sheet as well, if needed (column K).

Further consistent errors can be specified here also, and by the use of appropriate formulae, checks for entry of solutions to fewer decimal places could also be added (possibly with the use of an IF statement, to ensure that the student has indeed entered their solution to the relevant number of decimal places).

Both types of common errors can have marks attached to them, and these are specified in the column headed Marks for that error block.

As with consistent error checks, formulae can be quite complex, and here the possibilities are numerous for common error checks (including entering solutions into the wrong answer boxes, as well as the more obvious incorrect calculations that may be performed). The difficult part, beyond formulation of the Excel functions, is determining the many ways that a student could go wrong. This will be quite time-consuming!

Note that there is no need to specify the validity checks first in the row of errors, as all will be cycled through and validity checks performed irrespective of whether one of the other errors has been identified. The order of the other errors may be important, since once a common error has been identified, the others are not checked. However, since it is likely that only one will relate to the incorrect solution, in practice this may not be an issue.

Note also that when the program is run, the check for common errors is performed **before** the check for a consistent error (see Section 4.3). If a common error is found, the consistent error check is not performed. This fact may affect the best way to specify expected errors in the program, depending on the assessment.

4. e-Mark – running the program

4.1 Introduction

Running the e-Mark program involves clicking on each of the numbered buttons in turn on the MASTER sheet. The first step is to read in the student's solutions sheets. These will either be pre-formatted Excel sheets, or text files created from Word forms (see Section 5.3 for details of how to process the Word forms to create text files). A separate button number 1 (Read in all student solution sheets) exists for each of the two formats – press only the one relating to the relevant format.

On pressing this button, either the macro OpenAllText or OpenAllExcel is run (see Appendix B for code). The macro looks for a folder called Files which **must** be present in the same directory which the e-Mark file is residing in. It opens all relevant files that are in this folder and loads them into the e-Mark workbook. Note that these macros specify the name of the e-Mark workbook and so if the name is changed, it must also be changed here in the code!

4.2 Possible problems

A few points to bear in mind are: the code has not been written to ensure that the macros will run properly if other Excel workbooks are open at the same time. You must therefore close all other Excel workbooks before using e-Mark.

Also, there are no checks performed as to whether multiple copies of sheets for the same student have been read into the program. A check for this should be performed before the sheets are read in, and only one copy retained. If there are multiple sheets for a student, then only the one located closest to the right of the workbook will be the one that appears in the MASTER sheet post-marking (as the others will be overwritten), though all will be formally marked as the program proceeds through each sheet in turn.

When the marking macro is run (see below, in the next section, for details) a common problem that is encountered is that a student has not entered their ID number correctly on their answer sheet. It is best if these errors can be checked in advance of the files being loaded into the program, but the program does flag the problem when it finds it, and exits. The problem sheet is the current one, which can be noted down as to its number, the proper ID number found on the STUDENTS sheet, and copied into the relevant student's solutions sheet in the workbook. However, the marking macro starts anew at the left-most sheet each time, and so some time is wasted at the marking stage when encountering these problems.

4.3 Marking the work, with feedback

The actual marking of the work is performed when button 2 (Mark all student solution sheets) is pressed. This activates the MarkSheets macro (see Appendix B for code), which as a first step asks whether the feedback is via Word mail merge (to quickly determine the type of solution sheet and hence placement of feedback on those sheets). The worksheet screen is then minimised.

The macro proceeds by stepping through each sheet in turn, from left to right in the workbook. It first locates the student's ID number in the STUDENTS sheet, finding the relevant datasheet number for that student, and copying the relevant data into column C on the DATA sheet. Then it copies the correct answers for that student into column P of the SOLUTIONS sheet, and copies the student's answers into column Q. The student's dataset number is copied to their solution sheet for

future use (in cell A5000, hence there is a natural limit on the number of rows that can used for a pre-formatted Excel solution sheet). It also begins to format the MASTER sheet according to the number of questions, placement of intermediate marks and so on. Recall that at the start before the macro is run, the sheet is blank below row 4.

Marking then proceeds for that student's set of answers. First, it is determined whether an answer was submitted by the student, or whether the cell is empty. It is important that the program checks this specifically, as a blank cell in Excel is interpreted as a 0, which may of course be the correct answer in some cases! If the answer is missing, a cross is recorded, and the generic feedback of the correct answer provided (the correct answer feedback is "The correct answer is …" followed by the answer, whether that is numerical or text). In such a case, #VALUE! is printed to column Q instead of the student's blank answer, so that consistent error calculations will not pick up the blank as a 0, but will instead also be recorded as #VALUE!, meaning that a consistent error check does not make sense in this case.

For an actual answer, it is then determined whether the student's solution is correct or not (to within the specified tolerance if it is a numerical answer). If it is correct, a check is then made as to whether the answer must nevertheless satisfy a consistent error check. If it must, and it does not match the consistent error determined by the code in column K on the SOLUTIONS sheet, then its status is changed to being incorrect, the marks are set to zero, and appropriate feedback is recorded ("Although it matches the solutions, your answer is NOT consistent with earlier errors, and so is wrong."). If no consistent error check is being forced, then the answer remains correct, and no feedback is provided (note: feedback is only given for incorrect answers). In this case, a tick is recorded, and full marks given. In the previous case of failing a consistent error check, then a cross is recorded, and no further marking is done for that answer.

For an incorrect solution, the program proceeds to check for the reason, whether that is a consistent error or common error. It first performs an alternative decimal place check, if one has been specified on the SOLUTIONS sheet, to see whether the actual answer rounded to that many decimal places matches the student's solution, which it checks has been rounded to the same number of places. If this is so, then a tick is recorded, full marks given, and no further marking is done (no feedback is given). Note that a further check of a forced consistent error is **not** performed.

Next, a check for common errors takes place (recall that common errors are checked **before** the consistent error is checked – if a common error is found, there is **no** checking for consistency, as the reason why the answer is wrong has already been found). For common error checking (including a validity check), this only occurs if it has been declared on the SOLUTIONS sheet. In this case, the relevant number of errors to check, as given on the ERROR sheet, are tested in turn, with the values being read from the ERROR sheet and compared to the student's solutions. Where the student's solution is less than the stated minimum or greater than the stated maximum (i.e. failing the validity check), a cross is recorded and appropriate feedback given, such as "Your answer is greater than the possible maximum value of ", where the actual value is placed at the end of the sentence. However, further marking is performed here, to see whether another common error can be identified and additional feedback given. If it cannot, then the feedback is supplemented with the generic feedback of "The correct answer is …". If another common error is identified, then the appropriate feedback is added to the end of the validity check feedback. In either case, the marks awarded are those relating to the validity check failure, as per the number of marks specified on the ERROR sheet, since this is seen as a more serious error.

All common error feedback, except for validity checks, is taken from the relevant feedback column for that error on the ERROR sheet. When a common error is identified, no further checks of

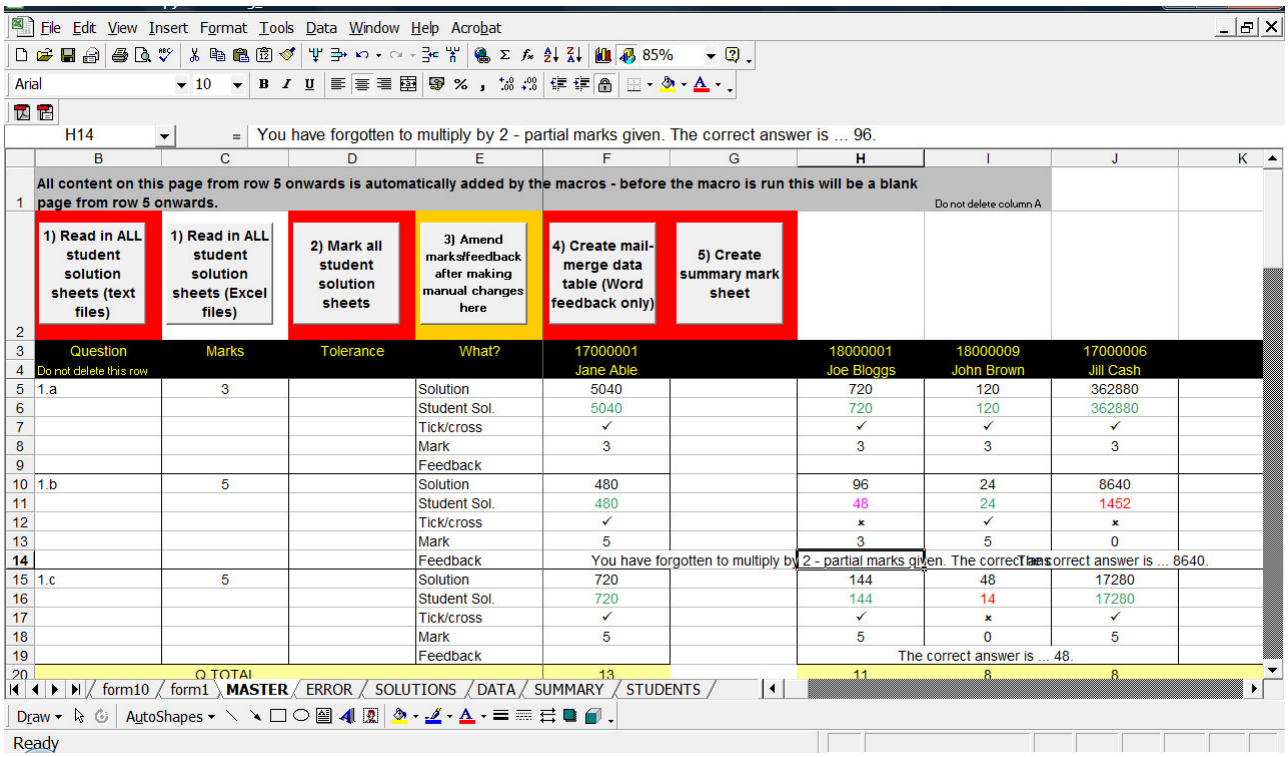
common errors are performed, except those for validity of an answer. This makes sense, as there should only be one way to identify the error made. Common errors are recorded as crosses, with marks taken from the ERROR sheet. Common error feedback is supplemented with “The correct answer is …”.

Only if no common errors are identified, and the validity checks are not failed, is a final check made to see if the student’s incorrect solution matches the value determined to be consistent with other solutions given by the student. In this way, it can be determined if the student’s solution, although incorrect, is nevertheless consistent with errors, and therefore the method has been performed correctly. For numerical solutions, the student’s answer must be within the question’s specified tolerance of the consistent error to be classified as satisfying this check. If it is satisfied, then a cross is still recorded, but the generic feedback of “Answer is incorrect, but it is consistent with other errors.” is given, together with the marks specified for a consistent error on the SOLUTIONS sheet.

All information (i.e. tick/cross, feedback) is placed on the MASTER sheet, as well as being copied to the student’s solution sheet. For ease of checking, as well as showing the actual solutions the student solutions are colour coded on the MASTER sheet according to the outcome of the above steps. Colour codes are

Green	Correct
Light green	Correct when marked to alternative decimal place
Amber	“Correct”, but fails check for consistency, therefore incorrect
Lilac	Incorrect, failed validity check
Pink	Incorrect, common error identified
Plum	Incorrect, failed validity check due to identified common error
Blue	Incorrect, but consistent with other errors
Red	Incorrect, no reason known

An example can be seen below, in the screenshot that was first shown in Chapter 1.



Note that on this screen the feedback that is placed in the student's column often runs into the next column when viewing, which makes it hard to read what is given. However, clicking on the relevant cell will mean that the full content can be seen in the formula bar at the top of the spreadsheet.

4.4 Auditing the work and human intervention

It should be remembered that e-Mark is only a basic tool, and results need to be audited to make sure that errors have not occurred in the marking process, either as a result of a bug in the code or because of problems with setting up the workbook for the assignment. The MASTER sheet is particularly useful for checking such things, since all student solutions (appropriately colour coded) are printed here (see Section 4.3 for an example screenshot). A quick scan can be made, comparing the student solution to the actual solution, to check that green solutions truly are correct, and red ones are truly incorrect. It may be the case that the student has used too few decimal places when entering their solution, and that the alternative decimal place marking has not been implemented. Or they have possibly swapped two digits when entering their answer – you may want to award marks in this case, or not. The colour coding is a useful aspect since it can be seen quite easily what has caused the incorrect solution (based on the common and consistent error statements you have set up in advance).

Where there are complex questions, it is often useful to ask the students to hand in their workings, so that incorrect answers can be looked at in detail (correct ones need not be, therefore still saving time marking in full). Looking at workings is likely to show where a student has gone wrong, and then the decision can be made whether to award partial marks – marks can be updated and appropriate feedback manually added to the MASTER sheet (this does not need to be detailed, since such feedback can be written on the pages of workings, but should instead indicate briefly the problem, and say that partial marks have been awarded).

Of course, there will also be situations when your assignments include graphical output or discursive passages that cannot be marked by computer. You will need to mark these by hand, and add the marks to the program yourself afterwards.

It is a simple matter to amend the results of the program. All components in a student's column on the MASTER sheet, apart from the actual and student's solutions, can be changed and copied onto the student's sheet. That is, the mark can be amended, a tick and cross can be interchanged, and the feedback can be changed, added or deleted.

For example, if for a 2-mark question it was decided that the student's solution of 0.7 for a question with actual solution 0.682 is acceptable, then the mark of 0 would need to be manually changed to 2 in the relevant cell on the MASTER sheet, the cross would need to be replaced by a tick (choose Insert>Symbol and select a Wingdings tick), and the feedback that would say "The correct answer is … 0.682" would either be completely deleted (by pressing the Delete button on the keyboard while that cell is highlighted) or perhaps replaced with new feedback (typed into the cell) saying something like "Answer correct to 1 decimal place – use more in future." If it was decided that the answer is not fully correct, as the question asked for 3 decimal places and precision has been lost, then the course of action may be to change the mark to 1, leave the cross in place, and change the feedback to "Answer correct to 1 decimal place, but question asked for 3. Partial marks given. The correct answer is … 0.682." Colour codes can be changed in the usual way, by clicking on the font colour icon or selecting Format>Cells>Font.

There is also the issue of different rounding causing problems with the program picking up consistent errors and awarding marks (Section 1.5). Recall that there may be a situation in which the students has approached the question incorrectly, and obtained errors for earlier questions. These answers are then needed in later questions, for example the ratio of two of them. The consistent error check will simply calculate the ratio of the two numbers the student has submitted on their answer sheet. These numbers will probably be rounded from the originals, and so the consistent error calculation will differ from what the student has entered, even though they have done the correct division. These cases are difficult to avoid in the program, and need to be addressed at the time the different datasets are generated. However, maual adjustment post marking is possible, for example if it can be seen that the two earlier values are either consistent errors, or common errors (blue or pink coded in the MASTER sheet), and the student would have used a computer to obtain the next answer, then it can be assumed quite reasonably that this later answer is indeed consistent with the earlier ones (provided it isn't wildly different of course!) A change can be made to the marks, tick/cross, and feedback changed, added or deleted as appropriate. Overall though, it may be better **not** to force consistent error checks for numeric solutions, unless is is known that rounding problems will not occur.

Note: if feedback is to be deleted, then it should be properly deleted, and not replaced with a space in the cell, since a space will be interpreted as being a piece of feedback later on, and will be assigned a number.

You can add up to two pieces of overall feedback here if you like (Feedback1 is usually used for denoting if the student is assumed to have indicated learning difficulties requiring sympathetic marking, e.g. no punishment for poor spelling or grammar, and would have been filled in automatically if the "green sticker" flag was set on the STUDENTS page). Feedback2 (or Feedback1 if not used) might include a criticism for not following the guidelines, or it may include a positive final statement about the quality of the work.

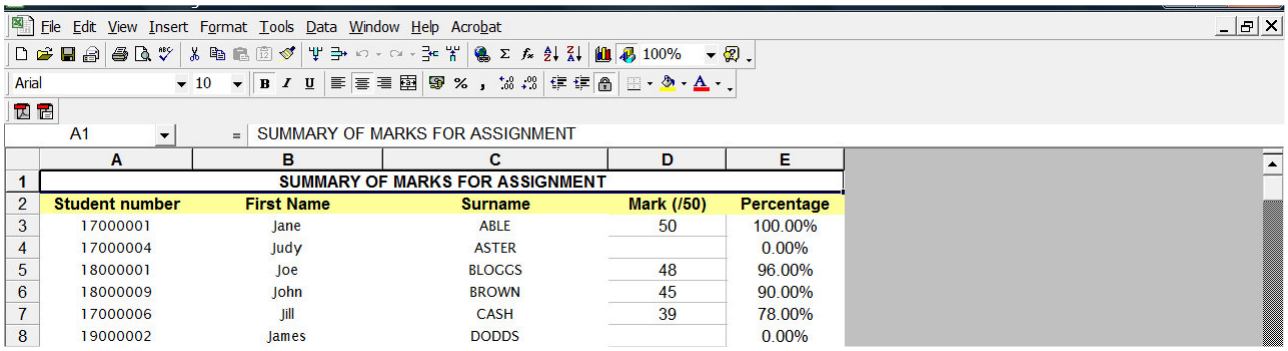
As many changes as you want can be made to the MASTER sheet, and there is no need to update the intermediate or total mark counts. These will be updated automatically when button 3 (Amend marks/feedback after making manual changes here) is pressed. This button **must** be pressed after all manual updates to the MASTER sheet have been completed, since this button executes the ChangeMarks macro (see Appendix B), which copies the marks, ticks/crosses and feedback to the student's solution sheets. It also performs a check that the number of marks awarded to a student for each question does not exceed the number allocated for that question. Note that running the MarkSheets macro again (button 2) overwrites any changes made to the MASTER and student solution sheets, so save often, or disable the macros to avoid you accidentally clicking this button.

For work that is to be audited by another, the e-Mark MASTER sheet can appear daunting, and the auditing process is likely to be complicated by the fact that each student has a different datasheet, therefore different expected solutions. It is advisable to print out the marked and updated student solution sheets (ready to be returned to the students), and to pass these on to the auditor, together with a copy of the assignment, outline generic solutions, any student workings, and also the e-Mark electronic file.

4.5 Summaries of solutions and marks

As noted above in Sections 4.3 and 4.4, the e-Mark program automatically provides a useful overview of the whole set of student solutions, plus actual answers, marks awarded, and feedback given. A copy of this spreadsheet can be taken and manipulated elsewhere to give an overview of marks per question for each student.

For a list of total marks, plus percentage, for the assignment, button 5 (Create summary mark sheet) should be pressed. This executes macro PrintMarks (see Appendix B) which copies the class list from the STUDENTS sheet, and pastes the mark in the Final Total row into the relevant cell in that list (the student with datasheet 1 is assumed to be in column F of the MASTER sheet, and so on). A student who has not handed in any work will have a blank total mark cell, but their percentage will be reported as 0.00%. This is demonstrated below.



A1 = SUMMARY OF MARKS FOR ASSIGNMENT

	A	B	C	D	E
1	SUMMARY OF MARKS FOR ASSIGNMENT				
2	Student number	First Name	Surname	Mark (/50)	Percentage
3	17000001	Jane	ABLE	50	100.00%
4	17000004	Judy	ASTER		0.00%
5	18000001	Joe	BLOGGS	48	96.00%
6	18000009	John	BROWN	45	90.00%
7	17000006	Jill	CASH	39	78.00%
8	19000002	James	DODDS		0.00%

Marks can be manually adjusted here, e.g. for adding results of any students for whom work had to be marked by hand, before printing and keeping in files.

Note that button 4 does not have to be pressed before button 5 (and in fact would not be pressed if feedback is not to be delivered by use of Word mail merge).

5. Word file for student solutions

5.1 Introduction

The e-Mark program has been designed to be flexible with the way in which students can hand in their solutions. The two obvious ways, since there needs to be an electronic copy for marking, is for them to enter solutions into an Excel sheet and either upload this to a VLE such as Blackboard, or email it to the lecturer, or alternatively enter solutions into a Word form, handing in a similar way via VLE or email. The Word form allows for more flexibility in terms of formatting, though possibly requires more set-up time, since a sheet is needed for both the feedback and student versions of the document.

5.2 Generating files for handing in solutions

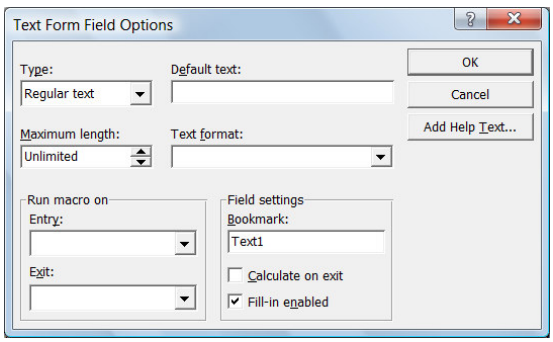
To produce a Word form for the students to download and enter their solutions into, the steps are straightforward. However, they are slightly different depending on whether the form is being created in Word 2003 or Word 2007 (in the latter case, the document should still be saved as an earlier version file, to allow for students who do not have the latest version). In either case, only basic information is given here, as this is a general Word procedure rather than being specific to the e-Mark program. However, you are advised to use simple forms rather than ActiveX controls.

You first need to create a Word file as usual, and type in the text for the solution sheet as you want it to appear. Since this is for students to enter their answers into, this is likely to be slightly different from the version that is given out as the assessment, but may be the same. Flexibility means that you can have a paragraph of text with gaps for solutions embedded within it, as well as tables with gaps. You should not have any preamble text at the top of the document, since once it is turned into a form and protected, the cursor will jump straight to the first box, bypassing any text. At the top of the page you should include a space for Name and Student ID Number.

Word 2003

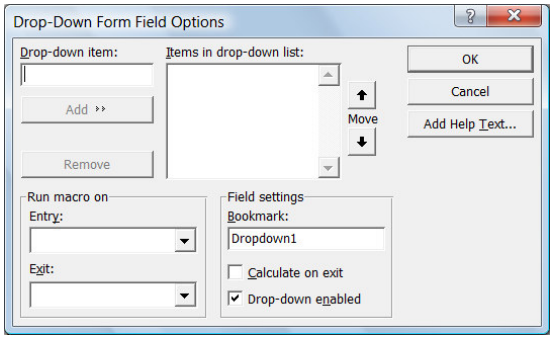
In Word 2003 or 2000, go to View>Toolbars and select Forms. The Forms toolbox will appear on the main toolbar at the top of the page, thus

The first icon represents a text box, and this allows for entry of text or numerical solutions, which will be the most common type of solution. The second icon is a check box, and the third a drop-down box of possible solutions that solutions can select one of (this is likely to be used often as well). On clicking the first icon, a box will be entered into the document and properties of it can be fixed by right-clicking on the box and selecting Properties, obtaining



For numerical solutions, it is useful to specify type as Number and then fix the number of decimal places that can be used. However, please note that Word truncates answers with more than the specified number of decimal places – it does not round!

For the third icon, right-clicking on the box that it produces gives



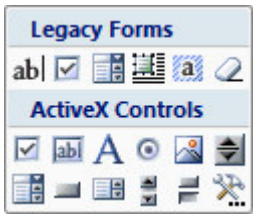
allowing for entry of each item in the drop-down list. The first entry, which is displayed by default before selection on entry, should not be one of the actual answers! It may be a simple …… or combination of answers, e.g. yes/no where the answers are yes and no. Note that there is a limit to the number of entries possible in the list.

Note: if the automatic grammar/spell check is on, Word may not allow you to right-click and select Properties until you have chosen to ignore the grammar/spelling error that it has highlighted.

Once the form has been created, you need to protect it so that the student cannot edit it, but can only enter their solutions into the relevant boxes. Select Tools>Protect Document then select Forms, and add a password. You can now make the form available to students to download from a VLE, or email it to them.

Word 2007

In Word 2007, the procedure is similar: go to the Developer ribbon (you may have to activate this if you cannot automatically see it), and hover over the forms icon so that the following menu appears



You only need the Legacy Forms icons. As for Word 2003, the first icon represents a text box, and this allows for entry of text or numerical solutions, which will be the most common type of solution. The second icon is a check box, and the third a drop-down box of possible solutions that solutions can select one of. On clicking the first icon, a box will be entered into the document and properties of it can be fixed by right-clicking on the box and selecting Properties – see the discussion for Word 2003 for details.

Once the form has been created, you need to protect it so that the student cannot edit it, but can only enter their solutions into the relevant boxes. In Word 2007 select Protect Document from the Developer Ribbon, and you will see the following panel at the right of your window

Check the second box, and select Filling in forms, thus

Click on Yes, Start Enforcing Protection, and you will be prompted for a password. You can now make the form available to students to download from a VLE, or email it to them. Note that you may want to save it as Word 2003, in case some students do not have Word 2007.

General issues

Note that the e-Mark program assumes that the very first form box contains the student's name, and the second box contains the student's ID number (which will be used to locate their dataset in the list). It is important then that these two boxes are placed in this order at the top of the form, else the e-Mark program will not run.

5.3 Processing the student files

Students can either hand in their completed Word forms by email, or by uploading to a VLE (e.g. using the Digital Dropbox or Assignments facility in Blackboard). Once these have been downloaded, they need processing to extract the information for reading into e-Mark.

The Word file Master.doc has been created to make the processing of these forms simple. This file contains a macro to search for all documents with extension .doc or .docx in a folder called Forms, open each one in turn, save only the form field content in a text file (labelled form1.txt onwards)

which it puts in a folder called Files, closing the documents afterwards. The code for this macro is in Appendix C. Note that for Word 2007 the Master file needs to be saved as a macro-enabled document, with extension .docm.

The file contains information about how to run the macro, which is called SaveAllForms. The Word screen is minimised while the macro is run, as it takes a few minutes to open e.g. 80 files and extract the information. Occasionally it fails to close a file, but these can be easily closed manually once it has finished running.

Note that this macro requires that the student Word forms are in a folder called Forms which is located in the same place that the Master.doc document is, and a folder called Files must have already been created in the same place, ready for receiving the text files. If the macro fails to run correctly, it may be a problem with one of these two folders not being present.

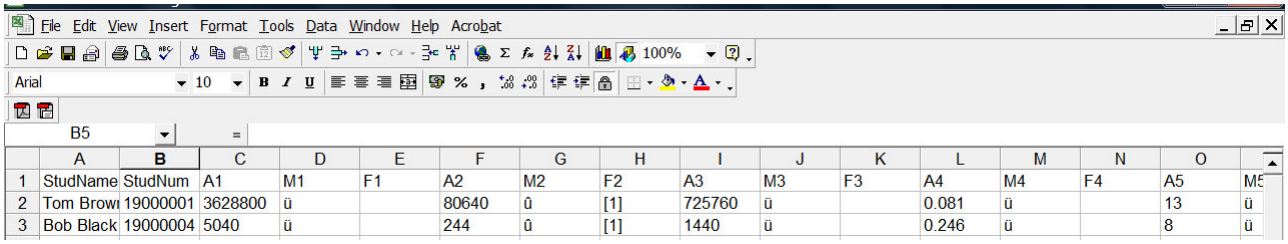
The text files that are saved in Files are now ready to be imported into e-Mark at the click of a button (see Section 3.1). However, occasionally the resulting text file has an errant linebreak in it, which means that it is not read correctly into e-Mark – such problems need to be looked for and corrected before commencing marking, else that sheet will not be marked correctly.

5.4 Generating files for returning marked work

Students ideally should have a paper copy of their marked work returned to them, as they will probably want their work in this medium when it comes to revision. The e-Mark program allows for this by producing feedback and ticks/crosses that can be printed to the sheet and returned to the student. For Word format feedback sheets, this is achieved through using the mail merge facility. Therefore, a template needs to be produced and linked into the source database.

5.4.1 Data source

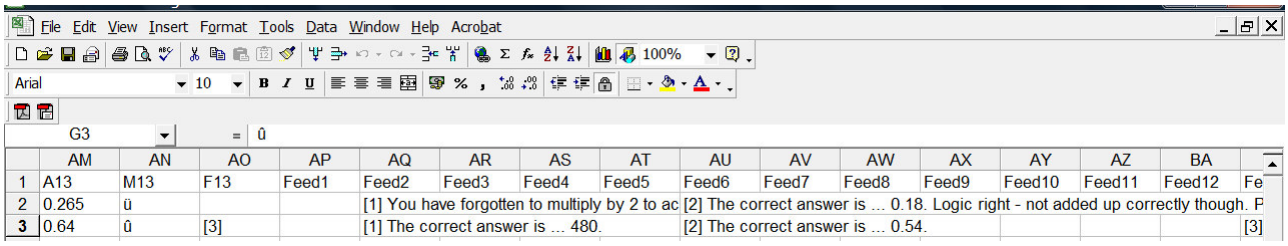
The source database will usually be an Excel table – this table is produced by e-Mark on pressing button 4 (Create mail-merge data table). A new Excel file called merge.xls (2000 or 2003 version of the program) or merge.xlsx (2007 version of the program) is created in the same folder that the e-Mark program resides in. It will look similar to



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	StudName	StudNum	A1	M1	F1	A2	M2	F2	A3	M3	F3	A4	M4	F4	A5
2	Tom Brow	19000001	3628800	ü		80640	û	[1]	725760	ü		0.081	ü		13
3	Bob Black	19000004	5040	ü		244	û	[1]	1440	ü		0.246	ü		8

The first row contains labels for each of the columns – these are the field names for inserting into the Word mail-merge document. A denotes the student's answer (1 … denotes the number of the question/answer). M denotes the marking symbol: when in Wingdings font these appear as ticks (ü) or crosses (û). F denotes the number of the feedback – here it assumed that the document will be formatted in such a way such that by each wrong answer, a number in superscript and brackets [ ] will appear, referring to a list of feedback given at the very end of the document.

Scrolling along in this file reveals, after all A M F values have appeared,

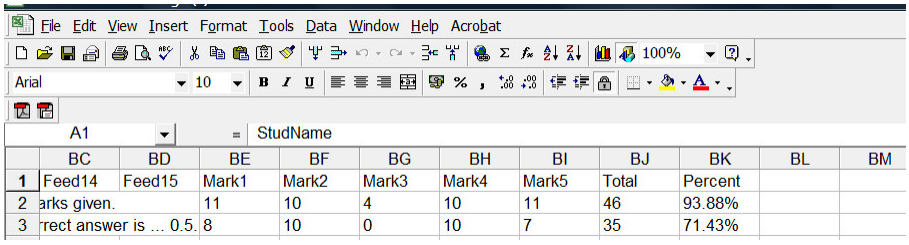


	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA
1	A13	M13	F13	Feed1	Feed2	Feed3	Feed4	Feed5	Feed6	Feed7	Feed8	Feed9	Feed10	Feed11	Feed12
2	0.265	ü			[1] You have forgotten to multiply by 2 to ac				[2] The correct answer is ... 0.18. Logic right - not added up correctly though. P						
3	0.64	û	[3]		[1] The correct answer is ... 480.				[2] The correct answer is ... 0.54.						[3]

Feed denotes the actual feedback provided (to be printed at the end of the document, as mentioned above). By default this includes the number relating to the piece of feedback, so that it is properly referred to. A linebreak character is also added to the end, to aid proper formatting of the mail-merge document when it is produced.

Note that there is a Feed column for every question, and if feedback is required for that question, it is printed in the relevant column. If the answer was correct, the cell for that question's feedback is empty. Empty cells are hidden by the text in the cell before it overrunning on the screen (as can be seen above). In order to properly read what is in a cell, click on that cell, and the contents will be displayed at the top of the screen, in the cell contents box, as usual.

Scrolling along still further reveals that there are two additional Feed columns (numbered after the final question, so in the example above, with 13 questions, the final two Feed columns are Feed14 and Feed15). These are for the two additional feedbacks that can be given, that were mentioned earlier (such as for informing that marking is consistent with a disability sticker, or other general observations that the lecturer wants to make). Finally, there are columns for marks.



	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM
1	Feed14	Feed15	Mark1	Mark2	Mark3	Mark4	Mark5	Total	Percent		
2	arks given.		11	10	4	10	11	46	93.88%		
3	rect answer is ... 0.5.		8	10	0	10	7	35	71.43%		

In this example, there were 5 questions, and so 5 intermediate marks are printed to the document. The values to be printed appear in the columns labelled Mark. The overall total is printed in the Total column, and the overall percentage in Percent.

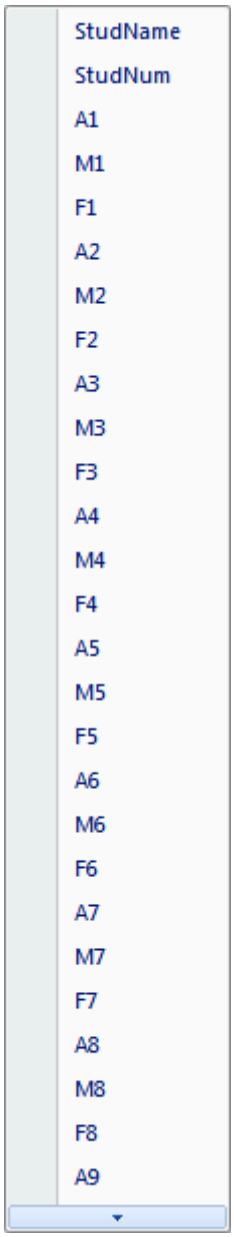
A template Excel file can either be set-up manually in this format at the start, or the e-Mark program can be run once with e.g. a single set of student's solutions to generate the file, and its corresponding set of headers for use as form fields in the mail merge document.

Note that Word 2007 requires the data source to be linked in before fields can be specified, whereas earlier versions of Word allow names to be specified without having already linked in a source. To ensure no errors with names of fields, it is probably better to always generate the file first, and link it in, no matter which version of Word is being used to generate the document.

5.4.2 Mail merge document

The mail merge document is set up in the same way that any mail merge would be, so you should consult the Help system for details. First select the option from the menu for mail merge (e.g. Start Mail Merge in Word 2007), and then link in the merge.xls? file as the data source – you should have already created the Word file how you want it to look in general, without the fields in place. Then you can proceed through the file, inserting mail merge fields into the document at the relevant

places, e.g. in Word 2007, selecting Insert Merge Field from the Mailings ribbon, the following list of possible merge fields appears



from which the relevant one can be selected. Note that these fields only appear because the merge.xlsx file has been selected as being the source of the data (through Select Recipients, and Use Existing List in Word 2007).

The suggested layout is the following: for each solution, first insert the relevant A field (for the student's solution), then insert the relevant M field (for the tick/cross), and then insert the relevant F field (for the feedback number). Fields are enclosed with << and >>, thus you may have

The average of the numbers is «A1» «M1» «F1»

as an initial entry.

Note though that the ticks and crosses will only appear properly if their font is Wingdings, and so the «M1» part needs to be formatted with this font. Change the font as usual in Word. Not also that

the feedback number needs to be superscript, hence highlight the relevant field and make this so. The sentence would now read in the mail merge document

The average of the numbers is «A1» ★💣🗁🕓 «F1»

Better formatting can be obtained by choosing to put the tick/cross and superscript after a full stop if they come at the end of a sentence, and so on. A few trial runs of creating the mail merge files will reveal the best way to format the document.

At the top of the document the fields for student name and number are required. These can be added in the same way as e.g. for A1 above (no special formatting is needed). Similarly, fields for intermediate and total marks can be added at the relevant places.

For the actual feedback, these fields should all be placed at the end of the document. There should be no spaces between them, since if the field is empty in the merge.xls? file, you will not want a space to nevertheless be printed, since this will lead to different formatting for each student, depending on the number of pieces of feedback they have. You will also not want each feedback field to appear on a separate line, since a blank line will then be present even if there is no feedback to be printed. Therefore, putting all of the feedback "Feed" fields next to each other, with no spaces, the desired appearance is obtained, where each piece of feedback appears on a new line, but no other lines or spaces appear. Therefore, at the end of your document you may have something like

«Feed1»«Feed2»«Feed3»«Feed4»«Feed5»«Feed6»«Feed7»«Feed8»«Feed9»«Feed10»«Feed11»«Feed12»«Feed13»«Feed14»«Feed15»

Once the template is set-up, the final mail merge can be run once the merge?.xls has been generated from the final version of the e-Mark program, after marking and auditing have been performed. This is achieved in Word 2007 by clicking on Finish & Merge.

A few trial runs of the final mail merge files are usually needed, to ensure a good location of page breaks for larger assignments. It may be necessary to run through the generated mail merge files to make small adjustments to pagination before printing, since different students will have different length files, depending on the number of feedback lines they have at the end.

Note that there is a limit to the number of fields that can be entered in a mail merge document when an Excel file is used as the data source – for large assignments with too many fields (you will find this out as in Word 2007 the drop-down list will not display the final fields) you will need to save the merge?.xls file as e.g. a text file, and link the text file into the document as the data source. You should be able to continue adding in fields then. However, note that you should save as a tab-delimited text file, not as an MS-DOS text file, since you need the tick and cross characters (ü and û) to be preserved in the file, and not replaced with unknown ones.

6. Problems and advice

There are undoubtedly going to be teething problems when using the program for the first few times, and some of these potential problems are highlighted below in our advice list. Some of these points have already been made, but are worth repeating here.

- Do not have any other Excel files open when the e-Mark or Datagen programs are run – they will fail.
- Keep a blank version of the program for changing each time when you create a new assessment – it is easier to do this than to delete the entries from the sheets, plus deleting the student sheets. However, you may want to first update the STUDENTS page before storing a "blank" version of the program, so you don't have to recreate this each time.
- Make sure that your STUDENTS sheet matches that in the Datagen program, or your own list of students and datasets – particularly if some students have left the course and you have updated your list midway through it.
- Generate your datasets carefully, checking that answers do not differ too much if students use different roundings – it will be hard to use e-Mark effectively if they do! If rounding makes quite a difference to answers, think about the tolerance carefully, and in particular, try to avoid forcing consistent errors for those numeric solutions that are affected.
- Check your solutions for each dataset carefully, particularly if you have generated them in a different program from that the students will be using (and, following on from the point above, noting that if the students are working through things by hand, rounding errors may mean that they obtain values very different to those a computer will produce – in such cases, it may be better to use a program to generate different sets of solutions by forcing rounding at each intermediate stage, and basing your tolerance on the difference seen between the solutions).
- Many problems may be encountered with students handing in wrongly formatted Word or Excel sheets of their solutions (e.g. they may use OpenOffice). Some of these can be spotted initially by checking file sizes in the folder that they have been downloaded to (2007 versions are usually about half the size of 2003 versions, but wrongly formatted ones are often very much larger or smaller). You can either ask the student to resubmit a correctly formatted version, create the right version yourself from what they have submitted, or print it out and mark those on paper.
- For Word submissions of solutions, check the text files that are produced either before or after loading into Excel, to make sure everything appears on one row (sometimes problems occur with saving to a text file and a linebreak is added, which causes problems in the auto-marking program which expects everything to be on one row). These can be fixed easily yourself before running the program, by cut and pasting the material where it should go.
- Check the names that the students have entered onto their solution sheets, as these will not be corrected if wrong. Although they won't appear in the final list of marks, they will remain on the marked versions handed back to students.
- Possibly check the student numbers as entered also, but those that can't be found in the list will be flagged later on when the program is run (though the program stops, and then has to start from the beginning again after the student number is fixed).
- For more complex assignments, it may not be appropriate that marking is completed by this program alone. Students may make intermediate errors that it is not possible to spot due to their final answer. In such cases, it may be appropriate to ask them to hand in on paper their workings, and then you can check only their wrong answers manually, making adjustments to the MASTER sheet if necessary based on what you see.

- Remember also – it may not be appopriate that a complex assignment is marked automatically, particularly if many different answers can be obtained through slightly different roundings. Think carefully whether using e-Mark is the correct thing to do.
- When it comes to auditing of the work, it may be helpful to make copies of the program, with different names, where a different marking strategy has been employed, e.g. forcing consistent errors in one, not in another, or allowing for different alternative decimal place marking. In this way different sets of solutions can be compared, and if one is acceptable for a particular question for a student, then its relevant formatted cell can be cut and pasted into the version of e-Mark deemed to be the final one, thus saving some time with manual editing.
- It is a good idea to highlight in colour anything on the MASTER sheet that has been manually changed, to keep a check of such changes. It is also desirable to note these changes down, so that an auditor can be made aware of them. Where errors have been identified that were not previously thought of, a note can be made to update the ERROR page next time.
- Do **not** click on button 2 after making any manual adjustment to the entries on the MASTER sheet – all changes WILL be overwritten!
- Make sure that the last row on the SOLUTIONS sheet has 0s entered for Solution Row and Solution Column, to indicate an intermediate total mark, else the final total will not be correct, as it adds up the intermediate totals.
- Make sure that marks are entered for consistent and common errors on the SOLUTIONS sheet, else they will be assumed to be 0.
- Tolerances for quantitative solutions will have 0.0000000001 automatically added to them to avoid problems with answers being exactly on the stated tolerance boundary, so make sure your tolerance is always bigger than this number, or your tolerance will be larger than you actually want.
- If any student solutions are in Excel 2007, then the e-Mark program should also be a 2007 version – there may be problems otherwise, as there is a difference in the number of columns on a worksheet between the versions, and the larger file will not be copied across to the e-Mark file.
- If possible, have a trial run with a small group of students, to help uncover the potential pitfalls of your assignment.
- **Most importantly, allow yourself plenty of time for planning and producing the datasets, solutions, and the e-Mark sheets!**

7. Possible future changes

The e-Mark program will be monitored and updates made as required. There may still be some undiscovered problems with the code, which will be fixed when highlighted. Improvements may include the ability to specify an acceptable alternative decimal place for each individual solution, instead of one to apply to the entire set of solutions. Also, an alternative decimal place may be added when checking for consistent errors.

At present only one consistent error can be specified for each question, though several common errors can be. If required, the program could be redesigned to allow for multiple consistent errors.

Also, there is currently no check made when checking for a consistent error, that any of the component parts of the formulae have indeed been entered incorrectly! While this is not a problem when the student has got the question wrong, if a consistent error is forced, and answers differ greatly according to the number of decimal places they have, then a correct solution may be overridden with an incorrect decision, because the consistent error (calculated on the basis of fewer decimal places than the actual answer) differs from the student's correct solution. A planned improvement to the program will be to check that earlier errors have indeed been made before forcing a consistent error check. This will be added in the first revision of e-Mark. Alongside this amendment will be the adding of a comment to each question label on the MASTER worksheet that indicates which rows are used in the consistent error calculation – this will make it easier to spot why an answer has failed a consistent error check for example, or why it has been flagged as one.

The program might be adaptable for student use for revision purposes. Questions could be set with several datasets available, allowing for repeated practice. Students could check their own solutions by running them through a version of the program.

Appendix A – Datagen code

Four different macros are provided to allow for the datasets to be provided in different formats, depending on the lecturer's choice. All are given here. The code assumes that the workbook is protected and the password stated below is `password` which is just for illustrative purposes. You should replace this with your own password that you use to protect the workbook, and the VisualBasic code.

```
' Declare global variables
Option Base 1
Dim TheData() As Variant ' data for printing
Dim WhichRow() As Integer ' which row to print to
Dim WhichCol() As Integer ' which column to print to
Dim NObs() As Integer ' number of observations for each variable (for writing to file)
Dim fs, f, fout
Const ForReading = 1, ForWriting = 2, ForAppending = 3
Const TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0

Sub PrintDatasheet()
'
' PrintDatasheet Macro
' Macro written July 2008 by Karen Ayres (updated from July 2005 version)
'

' Prints to pre-formatted DATASHEET in workbook

On Error GoTo Fatalquit ' If any run-time error occurs, exit painlessly from the program
                        ' with a warning, and reprotect workbook
GoTo ErrJump ' jump error handler
Fatalquit:
    ' Something has gone wrong - print error message and reprotect workbook
    MsgBox "An error has occurred in the program - please close it and start again. Make
sure this is the only Excel file you have open. If the problem occurs again, contact your
lecturer."
    Sheets("STUDENTS").Visible = False
    Sheets("DATA").Visible = False
    ActiveWindow.WindowState = xlMaximized
    Sheets("GENERATE").Protect ("password")
    ActiveWorkbook.Protect ("password")
    Exit Sub
ErrJump:
' First unprotect workbook

ActiveWorkbook.Unprotect ("password")
Sheets("GENERATE").Unprotect ("password")
ActiveWindow.WindowState = xlMinimized
Sheets("STUDENTS").Visible = True
Sheets("DATA").Visible = True

' Is Student on the list

StudentID = Range("C4").Value

If (Not (IsNumeric(StudentID)) Or StudentID <= 0) Then
    MsgBox "Your student number is not valid. Please re-enter and try again."
    GoTo EndThis
End If

' Try to find on the list

Sheets("STUDENTS").Select
Cells(1, 1).Select
Selection.End(xlDown).Select
NumStudents = Selection.Row - 1

For i = 1 To NumStudents
```

```
    If (Range(Cells(i + 1, 1), Cells(i + 1, 1)).Value = StudentID) Then
        Exit For
    End If
    ' Else continue
Next ' i

If (i > NumStudents) Then ' not on list
    MsgBox "Your student number could not be found on the list. Please re-enter and try
again, or contact the lecturer."
    GoTo EndThis
End If

StudentName = Range(Cells(i + 1, 2), Cells(i + 1, 2)).Value & " " & Range(Cells(i + 1,
3), Cells(i + 1, 3)).Value
DataSet = Range(Cells(i + 1, 4), Cells(i + 1, 4)).Value

Response = MsgBox("Are you " & StudentName & " ?", vbYesNo)
If Response = vbNo Then ' not the right person
    MsgBox "Please re-enter your student number and try again."
    GoTo EndThis
End If

' Can now print to DATASHEET

' Find out which cells to print to on each datasheet

' How many values need to be printed in total?

Sheets("DATA").Select
Cells(6, 6).Select
Selection.End(xlDown).Select
NumObs = Selection.Row - 5

ReDim WhichRow(NumObs + 1) As Integer
ReDim WhichCol(NumObs + 1) As Integer
ReDim TheData(NumObs + 1) As Variant

For j = 1 To NumObs
     WhichRow(j) = Range(Cells(j + 5, 6), Cells(j + 5, 6)).Value
     WhichCol(j) = Range(Cells(j + 5, 7), Cells(j + 5, 7)).Value
     TheData(j) = Range(Cells(j + 5, DataSet + 7), Cells(j + 5, DataSet + 7)).Value
Next 'j

Sheets("DATASHEET").Select
Range("H3").Value = StudentID
Range("H4").Value = StudentName

For j = 1 To NumObs
    Range(Cells(WhichRow(j), WhichCol(j)), Cells(WhichRow(j), WhichCol(j))).Value =
TheData(j)
Next 'j

MsgBox "Your data have been copied to the worksheet named DATASHEET in this workbook"

EndThis:
Sheets("GENERATE").Select
Range("A1").Select
Sheets("STUDENTS").Visible = False
Sheets("DATA").Visible = False
ActiveWindow.WindowState = xlMaximized
Sheets("GENERATE").Protect ("password")
ActiveWorkbook.Protect ("password")

End Sub
```

```
Sub PrintSheet()
'
' PrintSheet Macro
' Macro written July 2008 by Karen Ayres

' Prints to a new Excel sheet in workbook

On Error GoTo Fatalquit ' If any run-time error occurs, exit painlessly from the program
                        ' with a warning, and reprotect workbook
GoTo ErrJump ' jump error handler
Fatalquit:
    ' Something has gone wrong - print error message and reprotect workbook
    MsgBox "An error has occurred in the program - please close it and start again. Make
sure this is the only Excel file you have open. If the problem occurs again, contact your
lecturer."
    Sheets("STUDENTS").Visible = False
    Sheets("DATA").Visible = False
    ActiveWindow.WindowState = xlMaximized
    Sheets("GENERATE").Protect ("password")
    ActiveWorkbook.Protect ("password")
    Exit Sub
ErrJump:
' First unprotect workbook

ActiveWorkbook.Unprotect ("password")
Sheets("GENERATE").Unprotect ("password")
ActiveWindow.WindowState = xlMinimized
Sheets("STUDENTS").Visible = True
Sheets("DATA").Visible = True

' Is Student on the list

StudentID = Range("C4").Value

If (Not (IsNumeric(StudentID)) Or StudentID <= 0) Then
    MsgBox "Your student number is not valid. Please re-enter and try again."
    GoTo EndThis
End If

' Try to find on the list

Sheets("STUDENTS").Select
Cells(1, 1).Select
Selection.End(xlDown).Select
NumStudents = Selection.Row - 1

For i = 1 To NumStudents
    If (Range(Cells(i + 1, 1), Cells(i + 1, 1)).Value = StudentID) Then
        Exit For
    End If
    ' Else continue
Next ' i

If (i > NumStudents) Then ' not on list
    MsgBox "Your student number could not be found on the list. Please re-enter and try
again, or contact the lecturer."
    GoTo EndThis
End If

StudentName = Range(Cells(i + 1, 2), Cells(i + 1, 2)).Value & " " & Range(Cells(i + 1,
3), Cells(i + 1, 3)).Value
DataSet = Range(Cells(i + 1, 4), Cells(i + 1, 4)).Value

Response = MsgBox("Are you " & StudentName & " ?", vbYesNo)
If Response = vbNo Then ' not the right person
    MsgBox "Please re-enter your student number and try again."
    GoTo EndThis
End If

' Can now print to new worksheet
```

```
ThisRow = 6 ' start row

Sheets("DATA").Select

xxx = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value

For k = 1 To 1000 ' do as many times as there are questions
    If (IsNumeric(xxx) And xxx = 0) Then
        ' No more questions, so end
        GoTo EndThis:
    ' else continue
    End If

    ' create new excel sheet

    Set NewSheet = ActiveWorkbook.Sheets.Add

    ' How many columns for this question? (number of variables)

    Sheets("DATA").Select
    NumCols = Range(Cells(ThisRow, 3), Cells(ThisRow, 3)).Value

    For l = 1 To NumCols
        VN = Range(Cells(ThisRow, 4), Cells(ThisRow, 4)).Value
        NumObs = Range(Cells(ThisRow, 5), Cells(ThisRow, 5)).Value
        ReDim TheData(NumObs + 1) As Variant

        For j = 1 To NumObs
            TheData(j) = Range(Cells(ThisRow - 1 + j, DataSet + 7), Cells(ThisRow - 1 +
j, DataSet + 7)).Value
        Next 'j

        NewSheet.Select
        Range(Cells(1, l), Cells(1, l)).Value = VN
        For j = 1 To NumObs
            Range(Cells(j + 1, l), Cells(j + 1, l)).Value = TheData(j)
        Next 'j

        ThisRow = ThisRow + NumObs
        Sheets("DATA").Select
    Next 'l

    xxx = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value
Next ' k

EndThis:
If (k = 2) Then
    MsgBox "Your data have been printed to a new worksheet (probably named Sheet1)"
ElseIf (k > 2) Then
    MsgBox "Your data have been printed to new worksheets (one sheet per dataset,
probably named Sheet1 onwards)"
' else k = 1 so no questions were found!
End If

Sheets("GENERATE").Select
Range("A1").Select
Sheets("STUDENTS").Visible = False
Sheets("DATA").Visible = False
ActiveWindow.WindowState = xlMaximized
Sheets("GENERATE").Protect ("password")
ActiveWorkbook.Protect ("password")

End Sub
```

```
Sub PrintText()
'
' PrintText Macro
' Macro written July 2008 by Karen Ayres

' Prints to a text file (tab-delimited)

On Error GoTo Fatalquit ' If any run-time error occurs, exit painlessly from the program
                        ' with a warning, and reprotect workbook
GoTo ErrJump ' jump error handler
Fatalquit:
    ' Something has gone wrong - print error message and reprotect workbook
    MsgBox "An error has occurred in the program - please close it and start again. Make
sure this is the only Excel file you have open. If the problem occurs again, contact your
lecturer."
    Sheets("STUDENTS").Visible = False
    Sheets("DATA").Visible = False
    ActiveWindow.WindowState = xlMaximized
    Sheets("GENERATE").Protect ("password")
    ActiveWorkbook.Protect ("password")
    Exit Sub
ErrJump:
' First unprotect workbook

ActiveWorkbook.Unprotect ("password")
Sheets("GENERATE").Unprotect ("password")
ActiveWindow.WindowState = xlMinimized
Sheets("STUDENTS").Visible = True
Sheets("DATA").Visible = True

' Is Student on the list

StudentID = Range("C4").Value

If (Not (IsNumeric(StudentID)) Or StudentID <= 0) Then
    MsgBox "Your student number is not valid. Please re-enter and try again."
    GoTo EndThis
End If

' Try to find on the list

Sheets("STUDENTS").Select
Cells(1, 1).Select
Selection.End(xlDown).Select
NumStudents = Selection.Row - 1

For i = 1 To NumStudents
    If (Range(Cells(i + 1, 1), Cells(i + 1, 1)).Value = StudentID) Then
        Exit For
    End If
    ' Else continue
Next ' i

If (i > NumStudents) Then ' not on list
    MsgBox "Your student number could not be found on the list. Please re-enter and try
again, or contact the lecturer."
    GoTo EndThis
End If

StudentName = Range(Cells(i + 1, 2), Cells(i + 1, 2)).Value & " " & Range(Cells(i + 1,
3), Cells(i + 1, 3)).Value
DataSet = Range(Cells(i + 1, 4), Cells(i + 1, 4)).Value

Response = MsgBox("Are you " & StudentName & " ?", vbYesNo)
If Response = vbNo Then ' not the right person
    MsgBox "Please re-enter your student number and try again."
    GoTo EndThis
End If

' Can now print to file
```

```
ThisRow = 6 ' start row

Sheets("DATA").Select

xxx = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value

For k = 1 To 1000 ' do as many times as there are questions
    If (IsNumeric(xxx) And xxx = 0) Then
        ' No more questions, so end
        GoTo EndThis:
    ' else continue
    End If

    ' create new file

    TheFileName = Range(Cells(ThisRow, 2), Cells(ThisRow, 2)).Value

    If (IsNumeric(TheFileName)) Then GoTo Fatalquit

    TheFileName = ActiveWorkbook.Path & "\" & TheFileName

    Set fs = CreateObject("Scripting.FileSystemObject")
    fs.CreateTextFile TheFileName
    Set f = fs.GetFile(TheFileName)
    Set fout = f.OpenAsTextStream(ForWriting, TristateFalse)

    ' How many columns for this question? (number of variables)

    Sheets("DATA").Select
    NumCols = Range(Cells(ThisRow, 3), Cells(ThisRow, 3)).Value

    ' Have to store all data in an array before printing out row by row

    ReDim NObs(NumCols + 1) As Integer

    MaxNObs = 0
    For l = 1 To NumCols
        VN = Range(Cells(ThisRow, 4), Cells(ThisRow, 4)).Value
        NObs(l) = Range(Cells(ThisRow, 5), Cells(ThisRow, 5)).Value

        If (NObs(l) > MaxNObs) Then
            MaxNObs = NObs(l)
        End If

        ' write variable names to file now

        fout.write VN & Chr(9)
        ThisRow = ThisRow + NObs(l)
    Next 'l
    fout.writeline

    ' read in data

    ReDim TheData(NumCols + 1, MaxNObs + 1) As Variant

    ThisRow = 6

    For l = 1 To NumCols
        For j = 1 To NObs(l)
            TheData(l, j) = Range(Cells(ThisRow - 1 + j, DataSet + 7), Cells(ThisRow - 1
+ j, DataSet + 7)).Value
        Next 'j

        ThisRow = ThisRow + NObs(l)
    Next 'l

    ' can now print all variables to file, row by row

    For j = 1 To MaxNObs
```

```
        For l = 1 To NumCols
            If (Not (IsNull(TheData(l, j)))) Then
                fout.write TheData(l, j) & Chr(9)
            Else ' no data for this variable/row
                fout.write " " & Chr(9)
            End If
        Next 'l
        fout.writeline
    Next 'j

    xxx = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value
    fout.Close
Next ' k

EndThis:

If (k = 2) Then
    MsgBox "Your data have been printed to a text file, in directory " &
ActiveWorkbook.Path
ElseIf (k > 2) Then
    MsgBox "Your data have been printed to text files, in directory " &
ActiveWorkbook.Path
' else k = 1 so no questions were found!
End If

Sheets("GENERATE").Select
Range("A1").Select
Sheets("STUDENTS").Visible = False
Sheets("DATA").Visible = False
ActiveWindow.WindowState = xlMaximized
Sheets("GENERATE").Protect ("password")
ActiveWorkbook.Protect ("password")

End Sub
```

```
Sub PrintCSV()
'
' PrintCSV Macro
' Macro written July 2008 by Karen Ayres

' Prints to a CSV file

On Error GoTo Fatalquit ' If any run-time error occurs, exit painlessly from the program
                        ' with a warning, and reprotect workbook
GoTo ErrJump ' jump error handler
Fatalquit:
    ' Something has gone wrong - print error message and reprotect workbook
    MsgBox "An error has occurred in the program - please close it and start again. Make
sure this is the only Excel file you have open. If the problem occurs again, contact your
lecturer."
    Sheets("STUDENTS").Visible = False
    Sheets("DATA").Visible = False
    ActiveWindow.WindowState = xlMaximized
    Sheets("GENERATE").Protect ("password")
    ActiveWorkbook.Protect ("password")
    Exit Sub
ErrJump:
' First unprotect workbook

ActiveWorkbook.Unprotect ("password")
Sheets("GENERATE").Unprotect ("password")
ActiveWindow.WindowState = xlMinimized
Sheets("STUDENTS").Visible = True
Sheets("DATA").Visible = True

' Is Student on the list

StudentID = Range("C4").Value

If (Not (IsNumeric(StudentID)) Or StudentID <= 0) Then
    MsgBox "Your student number is not valid. Please re-enter and try again."
    GoTo EndThis
End If

' Try to find on the list

Sheets("STUDENTS").Select
Cells(1, 1).Select
Selection.End(xlDown).Select
NumStudents = Selection.Row - 1

For i = 1 To NumStudents
    If (Range(Cells(i + 1, 1), Cells(i + 1, 1)).Value = StudentID) Then
        Exit For
    End If
    ' Else continue
Next ' i

If (i > NumStudents) Then ' not on list
    MsgBox "Your student number could not be found on the list. Please re-enter and try
again, or contact the lecturer."
    GoTo EndThis
End If

StudentName = Range(Cells(i + 1, 2), Cells(i + 1, 2)).Value & " " & Range(Cells(i + 1,
3), Cells(i + 1, 3)).Value
DataSet = Range(Cells(i + 1, 4), Cells(i + 1, 4)).Value

Response = MsgBox("Are you " & StudentName & " ?", vbYesNo)
If Response = vbNo Then ' not the right person
    MsgBox "Please re-enter your student number and try again."
    GoTo EndThis
End If

' Can now print to file
```

```
ThisRow = 6 ' start row

Sheets("DATA").Select

xxx = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value

For k = 1 To 1000 ' do as many times as there are questions
    If (IsNumeric(xxx) And xxx = 0) Then
        ' No more questions, so end
        GoTo EndThis:
    ' else continue
    End If

    ' create new file

    TheFileName = Range(Cells(ThisRow, 2), Cells(ThisRow, 2)).Value

    If (IsNumeric(TheFileName)) Then GoTo Fatalquit

    TheFileName = ActiveWorkbook.Path & "\" & TheFileName

    Set fs = CreateObject("Scripting.FileSystemObject")
    fs.CreateTextFile TheFileName
    Set f = fs.GetFile(TheFileName)
    Set fout = f.OpenAsTextStream(ForWriting, TristateFalse)

    ' How many columns for this question? (number of variables)

    Sheets("DATA").Select
    NumCols = Range(Cells(ThisRow, 3), Cells(ThisRow, 3)).Value

    ' Have to store all data in an array before printing out row by row

    ReDim NObs(NumCols + 1) As Integer

    MaxNObs = 0
    For l = 1 To NumCols
        VN = Range(Cells(ThisRow, 4), Cells(ThisRow, 4)).Value
        NObs(l) = Range(Cells(ThisRow, 5), Cells(ThisRow, 5)).Value

        If (NObs(l) > MaxNObs) Then
            MaxNObs = NObs(l)
        End If

        ' write variable names to file now

        fout.write Chr(34) & VN & Chr(34)
        If (l < NumCols) Then
            fout.write ","
        End If
        ThisRow = ThisRow + NObs(l)
    Next 'l
    fout.writeline

    ' read in data

    ReDim TheData(NumCols + 1, MaxNObs + 1) As Variant

    ThisRow = 6

    For l = 1 To NumCols
        For j = 1 To NObs(l)
            TheData(l, j) = Range(Cells(ThisRow - 1 + j, DataSet + 7), Cells(ThisRow - 1
+ j, DataSet + 7)).Value
        Next 'j

        ThisRow = ThisRow + NObs(l)
    Next 'l
```

```
    ' can now print all variables to file, row by row

    For j = 1 To MaxNObs
        For l = 1 To NumCols
            If (Not (IsNull(TheData(l, j)))) Then
                If (IsNumeric(TheData(l, j))) Then
                    fout.write TheData(l, j)
                    If (l < NumCols) Then
                        fout.write ","
                    End If
                Else ' text, put in quotation marks
                    fout.write Chr(34) & TheData(l, j) & Chr(34)
                    If (l < NumCols) Then
                        fout.write ","
                    End If
                End If
            Else ' no data for this variable/row
                If (l < NumCols) Then
                    fout.write ","
                End If
            End If
        Next 'l
        fout.writeline
    Next 'j

    xxx = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value
    fout.Close
Next ' k

EndThis:

If (k = 2) Then
    MsgBox "Your data have been printed to a CSV file, in directory " &
ActiveWorkbook.Path
ElseIf (k > 2) Then
    MsgBox "Your data have been printed to CSV files, in directory " &
ActiveWorkbook.Path
' else k = 1 so no questions were found!
End If

Sheets("GENERATE").Select
Range("A1").Select
Sheets("STUDENTS").Visible = False
Sheets("DATA").Visible = False
ActiveWindow.WindowState = xlMaximized
Sheets("GENERATE").Protect ("password")
ActiveWorkbook.Protect ("password")

End Sub
```

Appendix B – e-Mark code

```
' VERSION: 16 Jan 09

' Declare global variables
Option Base 1
Public SolRow(400) As Integer ' which row on student worksheet that contains the answer
box per question
Public SolCol(400) As Integer ' which column on student worksheet that contains the
answer box per question
Public QOrTotal(400) As Integer ' indicator for each row, 0=Q, 1=total, -1=graph (marks
added manually)
Public Answer(400) As Variant ' the real answer per question
Public Tolerance(400) As Variant ' the acceptable limit of difference between the student
answer and the real one, per question
Public Marks(400) As Variant ' the available marks per question (can be fractional)
Public StudMarks As Variant ' marks the student has earned for current question
Public StudSols(400) As Variant ' the student's answer per question
Public ConsError(400) As Integer ' allow consistent error (1) or not (0)
Public ForceCE(400) As Integer ' force consistent error check (1) or not (0)
Public MarksCE(400) As Variant ' marks available for consistent error
Public CommError(400) As Integer ' check for common error (1) or not (0)
Public CommRow(400) As Integer ' row on ERROR sheet to look at
Public QLabel(400) As Variant ' label for question
Public CESol As Variant ' consistent/common solutions
Public Datasheets(400, 3) As Variant ' student IDs and their dataset number and
disability flag
Public GAR As Integer ' indicator for colour to highlight answer on master sheet
Public StudTotal(500) As Variant  ' overall mark for all students
Public RowsColsM(400, 2) As Integer ' places to print tick/cross marks on student sheets
Public RowsColsF(400, 2) As Integer ' places to print feedback on student sheets
Public RowsColsT(400, 2) As Integer ' places to print Q totals on student sheets
Public FeedRC1(2) As Integer ' places to print extra feedback 1
Public FeedRC2(2) As Integer ' places to print extra feedback 2
Public FeedBack(400, 2) As Variant ' feedback given, after manual updating
Public QTotal(100) As Variant ' Q totals, after manual updating
Public FeedBackText As String ' feedback given


Sub OpenAllText()
'
' OpenAllText Macro
' Macro written July 2008 by Karen Ayres
'
' Macro opens all files (expected to be text files) in the given directory
' and moves them into this workbook

xxx = MsgBox("Note: this function will NOT open files that are READ-ONLY. If you have any
files like this, click Cancel and change their properties before running this macro",
vbOKCancel)
If (xxx = vbCancel) Then Exit Sub

MyFilePath = ActiveWorkbook.Path & "\Files\"

MyFileToOpen = Dir(MyFilePath & "*.txt")

If MyFileToOpen = "" Then
    MsgBox "There were no files found - make sure they all have .txt extension and are in
subfolder named Files."
    Exit Sub
End If

Application.WindowState = xlMinimized
Do While MyFileToOpen <> ""
    MyFileName = MyFilePath & MyFileToOpen
    Workbooks.OpenText Filename:=MyFileName, Origin _
        :=xlWindows, StartRow:=1, DataType:=xlDelimited, TextQualifier:= _
```

```
        xlDoubleQuote, Comma:=True
    ActiveSheet.Select
    ActiveSheet.Move Before:=Workbooks("e-Mark.xlsm").Sheets(1)
    MyFileToOpen = Dir
Loop
Sheets("MASTER").Select
Application.WindowState = xlMaximized
End Sub
```

Sub OpenAllExcel()

```
'
' OpenAllExcel Macro
' Macro written July 2008 by Karen Ayres
'
' Macro opens all files (expected to be Excel files) in the given directory
' and moves them into this workbook

xxx = MsgBox("Note: this function will NOT open files that are READ-ONLY. If you have any
files like this, click Cancel and change their properties before running this macro",
vbOKCancel)
If (xxx = vbCancel) Then Exit Sub

MyFilePath = ActiveWorkbook.Path & "\Files\"

MyFileToOpen = Dir(MyFilePath & "*.xls?")

If MyFileToOpen = "" Then
    MsgBox "There were no files found - make sure they all have .xls or .xls extension
and are in subfolder named Files."
    Exit Sub
End If

Application.WindowState = xlMinimized
Do While MyFileToOpen <> ""
    MyFileName = MyFilePath & MyFileToOpen
    Workbooks.Open Filename:=MyFileName
    ActiveSheet.Select
    ActiveSheet.Move Before:=Workbooks("e-Mark.xlsm").Sheets(1)
    MyFileToOpen = Dir
Loop
Sheets("MASTER").Select
Application.WindowState = xlMaximized
End Sub
```

Sub MarkSheets()

```
'
' MarkSheets Macro
' Macro written October 2008 by Karen Ayres
'
' Cycles through the student worksheets in this workbook, picks up
' the student answers, checks them against the real ones, ticks or
' crosses against the answer on the student's sheet, and prints
' their answers and marks to the MASTER sheet

ActiveWindow.WindowState = xlMinimized
WhichType = MsgBox("Is this assignment using Word forms for solutions and feedback?",
vbYesNo)

' green  50 correct
' lt grn 4  correct to alt decimal places
' amber  45 technically correct, but not consistent with errors
' red    3  wrong
' blue   5  wrong but consistent with earlier error
' pink   7  wrong but due to common error specified
' purple 39 fails validity check
' plum   54 fails validity check and has a common error identified

NumSheets = Sheets.Count
```

```
'move non-student sheets to be last sheets if they aren't already

Sheets("MASTER").Select
Sheets("MASTER").Move after:=Sheets(NumSheets)

Sheets("ERROR").Select
Sheets("ERROR").Move after:=Sheets(NumSheets)

Sheets("SOLUTIONS").Select
Sheets("SOLUTIONS").Move after:=Sheets(NumSheets)

Sheets("DATA").Select
Sheets("DATA").Move after:=Sheets(NumSheets)

Sheets("SUMMARY").Select
Sheets("SUMMARY").Move after:=Sheets(NumSheets)

Sheets("STUDENTS").Select
Sheets("STUDENTS").Move after:=Sheets(NumSheets)

NumSheets = NumSheets - 6 ' number of student solution sheets

' first read in the locations for printing things to sheets

Sheets("SOLUTIONS").Select
FinalRow = Range("D1").Value
Range("I2").FormulaR1C1 = "=ISBLANK(R1C9)"
If (Range("I2").Value) Then ' no alternative decimal place checking
    AltDecCheck = 0
Else
    AltDec = Range("I1").Value
    If (Not (IsNumeric(AltDec))) Then
        AltDecCheck = 0
    ElseIf ((Abs(Int(AltDec) - AltDec)) > 0) Then
        ' not an integer
        AltDecCheck = 0
        MsgBox "Alternative decimal place entry not an integer - ignored"
    Else
        ' do use a different dec place for checking
        AltDecCheck = 1
    End If
End If
Range("I2").Clear

WhichQ = 0
WhichAns = 1
MaxCol = 0
FullMarks = 0
For i = 5 To FinalRow
    If (Range(Cells(i, 4), Cells(i, 4)).Value = 0) Then
        ' total for (part)question
        QOrTotal(i - 4) = 1
        WhichQ = WhichQ + 1
        If (WhichType = vbYes) Then
            RowsColsT(WhichQ, 1) = 5
            RowsColsT(WhichQ, 2) = 2 + WhichQ
        Else
            RowsColsT(WhichQ, 1) = Range(Cells(i, 6), Cells(i, 6)).Value
            RowsColsT(WhichQ, 2) = Range(Cells(i, 7), Cells(i, 7)).Value
        End If
    ElseIf (Range(Cells(i, 4), Cells(i, 4)).Value = -1) Then
        ' graph - not for marking
        QOrTotal(i - 4) = -1
        ' Question label
        QLabel(i - 4) = Range(Cells(i, 1), Cells(i, 1)).Value
        ' Marks available
        Marks(i - 4) = Range(Cells(i, 2), Cells(i, 2)).Value
        FullMarks = FullMarks + Marks(i - 4)
        ' feedback location
```

```
        If (WhichType = vbYes) Then
            RowsColsM(WhichAns, 1) = 2
            RowsColsF(WhichAns, 1) = 4
            If (Range(Cells(i, 6), Cells(i, 6)).Value = -1) Then
                If (WhichType = vbNo) Then
                    MsgBox "The SOLUTIONS sheet indicates that Word feedback WILL be used
- please re-run and select Yes when asked this question"
                    Exit Sub
                End If
                MaxCol = MaxCol + 1
            End If
        Else
            RowsColsM(WhichAns, 1) = Range(Cells(i, 6), Cells(i, 6)).Value
            RowsColsF(WhichAns, 1) = Range(Cells(i, 8), Cells(i, 8)).Value
        End If
        RowsColsM(WhichAns, 2) = Range(Cells(i, 7), Cells(i, 7)).Value
        RowsColsF(WhichAns, 2) = Range(Cells(i, 9), Cells(i, 9)).Value
        WhichAns = WhichAns + 1
    Else
        ' question (to be marked)
        QOrTotal(i - 4) = 0
        If (WhichType = vbYes) Then
            RowsColsM(WhichAns, 1) = 2
            RowsColsF(WhichAns, 1) = 4
            If (Range(Cells(i, 6), Cells(i, 6)).Value = -1) Then
                If (WhichType = vbNo) Then
                    MsgBox "The SOLUTIONS sheet indicates that Word feedback WILL be used
- please re-run and select Yes when asked this question"
                    Exit Sub
                End If
                MaxCol = MaxCol + 1
            End If
        Else
            RowsColsM(WhichAns, 1) = Range(Cells(i, 6), Cells(i, 6)).Value
            RowsColsF(WhichAns, 1) = Range(Cells(i, 8), Cells(i, 8)).Value
        End If
        RowsColsM(WhichAns, 2) = Range(Cells(i, 7), Cells(i, 7)).Value
        RowsColsF(WhichAns, 2) = Range(Cells(i, 9), Cells(i, 9)).Value
        ' Question label
        QLabel(i - 4) = Range(Cells(i, 1), Cells(i, 1)).Value
        ' Student solution locations
        SolRow(WhichAns) = Range(Cells(i, 4), Cells(i, 4)).Value
        SolCol(WhichAns) = Range(Cells(i, 5), Cells(i, 5)).Value
        ' Marks available
        Marks(i - 4) = Range(Cells(i, 2), Cells(i, 2)).Value
        FullMarks = FullMarks + Marks(i - 4)
        ' Tolerance
        Tolerance(WhichAns) = Range(Cells(i, 3), Cells(i, 3)).Value
        ' add tiny amount to avoid =tolerance problems
        Tolerance(WhichAns) = Tolerance(WhichAns) + 0.0000000001
        ' Consistent error information
        ConsError(WhichAns) = Range(Cells(i, 10), Cells(i, 10)).Value
        ForceCE(WhichAns) = Range(Cells(i, 12), Cells(i, 12)).Value
        MarksCE(WhichAns) = Range(Cells(i, 13), Cells(i, 13)).Value
        ' Common error information
        CommError(WhichAns) = Range(Cells(i, 14), Cells(i, 14)).Value
        CommRow(WhichAns) = Range(Cells(i, 15), Cells(i, 15)).Value

        WhichAns = WhichAns + 1
    End If
Next 'i

WhichQ = WhichQ + 1 ' overall total now
If (WhichType = vbYes) Then
    RowsColsT(WhichQ, 1) = 5
    RowsColsT(WhichQ, 2) = 2 + WhichQ
    FeedRC1(1) = 6
    FeedRC1(2) = 3
    FeedRC2(1) = 6
    FeedRC2(2) = 4
```

```
Else
    RowsColsT(WhichQ, 1) = Range("L1").Value
    RowsColsT(WhichQ, 2) = Range("N1").Value
    FeedRC1(1) = Range("L2").Value
    FeedRC1(2) = 1
    FeedRC2(1) = Range("N2").Value
    FeedRC2(2) = 1
End If

' write down flag for Q or Total value on MASTER sheet, column A
' also format sheet initially, for early columns

Worksheets("MASTER").Select
ThisRow = 5
WhichAns = 1
For i = 5 To FinalRow
    If (QOrTotal(i - 4) <= 0) Then ' not Q total
        Range(Cells(ThisRow, 1), Cells(ThisRow + 4, 1)).Value = 0
        Range(Cells(ThisRow, 2), Cells(ThisRow + 4, 2)).Select
        With Selection.Borders(xlEdgeLeft)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeTop)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        Range(Cells(ThisRow, 3), Cells(ThisRow + 4, 3)).Select
        With Selection.Borders(xlEdgeLeft)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeTop)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        Range(Cells(ThisRow, 4), Cells(ThisRow + 4, 4)).Select
        With Selection.Borders(xlEdgeLeft)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeTop)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
```

```
        End With
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        Range(Cells(ThisRow, 5), Cells(ThisRow + 4, 5)).Select
        With Selection.Borders(xlEdgeLeft)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeTop)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        Range(Cells(ThisRow, 5), Cells(ThisRow, 5)).Value = "Solution"
        Range(Cells(ThisRow + 1, 5), Cells(ThisRow + 1, 5)).Value = "Student Sol."
        Range(Cells(ThisRow + 2, 5), Cells(ThisRow + 2, 5)).Value = "Tick/cross"
        Range(Cells(ThisRow + 3, 5), Cells(ThisRow + 3, 5)).Value = "Mark"
        Range(Cells(ThisRow + 4, 5), Cells(ThisRow + 4, 5)).Value = "Feedback"

        Range(Cells(ThisRow, 2), Cells(ThisRow, 2)).Value = QLabel(i - 4)
        Range(Cells(ThisRow, 3), Cells(ThisRow, 3)).Value = Marks(i - 4)

        If (QOrTotal(i - 4) >= 0) Then
            If (Tolerance(WhichAns) = 0.0000000001) Then
                Range(Cells(ThisRow, 4), Cells(ThisRow, 4)).Value = ""
            Else
                Range(Cells(ThisRow, 4), Cells(ThisRow, 4)).Value = Tolerance(WhichAns) -
0.0000000001
            End If
        End If

        WhichAns = WhichAns + 1
        ThisRow = ThisRow + 5
    Else ' Q total
        Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value = 1
        Range(Cells(ThisRow, 3), Cells(ThisRow, 3)).Value = "Q TOTAL"
        Range(Cells(ThisRow, 2), Cells(ThisRow, 5)).Select
        With Selection.Interior
            .ColorIndex = 36
            .Pattern = xlSolid
        End With
        ThisRow = ThisRow + 1
    End If
Next 'i
Range(Cells(ThisRow, 3), Cells(ThisRow, 3)).Value = "FINAL TOTAL"
Range(Cells(ThisRow, 2), Cells(ThisRow, 5)).Select
With Selection.Interior
    .ColorIndex = 6
    .Pattern = xlSolid
End With
```

```
' now print two extra feedback rows

Range(Cells(ThisRow + 1, 5), Cells(ThisRow + 1, 5)).Value = "Extra Feedback 1"
Range(Cells(ThisRow + 2, 5), Cells(ThisRow + 2, 5)).Value = "Extra Feedback 2"

Range(Cells(ThisRow + 1, 2), Cells(ThisRow + 2, 2)).Select
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
Range(Cells(ThisRow + 1, 3), Cells(ThisRow + 2, 3)).Select
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
Range(Cells(ThisRow + 1, 4), Cells(ThisRow + 2, 4)).Select
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
Range(Cells(ThisRow + 1, 5), Cells(ThisRow + 2, 5)).Select
With Selection.Borders(xlEdgeLeft)
```

```
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With

' store values to be used in other macros

Range("C4").Value = FullMarks ' total marks available
Range("D4").Value = ThisRow ' final row on this MASTER worksheet
TotalRow = ThisRow

' read in list of student IDs and dataset numbers

Worksheets("STUDENTS").Select

StartRow = 2  ' row to start reading data from
Cells(StartRow, 1).Select
Selection.End(xlDown).Select
LastRow = Selection.Row

For i = StartRow To LastRow
    Datasheets(i - 1, 1) = Range(Cells(i, 1), Cells(i, 1)).Value ' ID
    Datasheets(i - 1, 2) = Range(Cells(i, 4), Cells(i, 4)).Value ' dataset
    Datasheets(i - 1, 3) = Range(Cells(i, 5), Cells(i, 5)).Value ' disability flag
Next 'i

' can step through sheets 1 to numSheets, knowing they are not non-student sheets

For ii = 1 To NumSheets ' one sheet per student
    Worksheets(ii).Activate

    ' assumes Student ID and Name are ALWAYS in B1 and A1
    StudID = ActiveSheet.Range("B1").Value
    StudName = ActiveSheet.Range("A1").Value

    ' find which dataset this student has

    dataset = 0
    For j = 1 To (LastRow - 1)
        If (StudID = Datasheets(j, 1)) Then
            dataset = Datasheets(j, 2)
            dbflag = Datasheets(j, 3)
            GoTo CarryOn
        End If
    Next 'j

CarryOn:

    ' check student ID is OK

    If (dataset = 0) Then
        MsgBox "ERROR - could not locate student ID " & StudID & " for " & StudName & "
in list - please fix and re-run marking macro"
        Exit Sub
    End If
```

```
    ' print dataset number to cell A5000 on student solution sheet (used later)

    Range("A5000").Value = dataset

    ' fill in feedback for disability sticker students

    If (dbflag = 1) Then
        Feed1 = "Your work has been marked assuming that you have attached a green
sticker to it, and appropriate allowance made according to the text on that sticker."
    Else
        Feed1 = ""
    End If
    Feed2 = ""

    ' copy student's dataset to column C

    Sheets("DATA").Select

    Range(Cells(3, dataset + 3), Cells(50000, dataset + 3)).Select
    Selection.Copy
    Range("C3").Select
    Selection.PasteSpecial Paste:=xlValues

    ' get real answers for this sheet, and copy to col P

    Sheets("SOLUTIONS").Select

    Range(Cells(5, dataset + 17), Cells(5000, dataset + 17)).Select
    Selection.Copy
    Range(Cells(5, 16), Cells(5, 16)).Select
    Selection.PasteSpecial Paste:=xlValues

    WhichAns = 1
    For xx = 5 To FinalRow
        If (QOrTotal(xx - 4) <= 0) Then ' Q/graph to mark (will be empty for graph)
            Answer(WhichAns) = Range(Cells(xx, dataset + 17), Cells(xx, dataset +
17)).Value
            WhichAns = WhichAns + 1
        End If
    Next 'xx

    ' now get student's actual solutions, and copy to col Q

    Worksheets(ii).Activate

    WhichAns = 1
    For xx = 5 To FinalRow
        If (QOrTotal(xx - 4) = 0) Then ' Q to mark
            StudSols(WhichAns) = Range(Cells(SolRow(WhichAns), SolCol(WhichAns)),
Cells(SolRow(WhichAns), SolCol(WhichAns))).Value
        End If
        If (QOrTotal(xx - 4) <= 0) Then
            WhichAns = WhichAns + 1 ' Q/graph to mark
        End If
    Next 'xx

    Sheets("SOLUTIONS").Select

    WhichAns = 1
    For xx = 5 To FinalRow
        If (QOrTotal(xx - 4) <= 0) Then ' Q/graph to mark
            If (Not (IsNumeric(StudSols(WhichAns)))) Then
                Range(Cells(xx, 17), Cells(xx, 17)).Value = "'" & StudSols(WhichAns)
            Else
                Range(Cells(xx, 17), Cells(xx, 17)).Value = StudSols(WhichAns)
            End If
            WhichAns = WhichAns + 1
        End If
    Next 'xx
```

```
    NumFeed = 0

    ' mark

    WhichAns = 1
    QRunSum = 0
    ThisRow = 5
    NewTotal = 0
    WhichCol = dataset + 5

    Sheets("MASTER").Select

    Range(Cells(3, WhichCol), Cells(3, WhichCol)).Value = StudID
    Range(Cells(4, WhichCol), Cells(4, WhichCol)).Value = StudName

    For xx = 5 To FinalRow
        If (QOrTotal(xx - 4) = 0) Then ' Q to mark
            WrongF = 0 ' flag for feedback or not
            ValidYN = 1 ' flag for valid answer (1) or not (0)
            CoErrNY = 1 ' flag for no common error (1) or common error (0)
            FeedBackText = ""

            ' add ' to avoid problems with text answer leading with an = sign
            ' then check whether there is a blank cell (i.e. no answer, rather than zero)

            If (Not (IsNumeric(StudSols(WhichAns)))) Then
                Range("A1").Value = "'" & StudSols(WhichAns)
            Else
                Range("A1").Value = StudSols(WhichAns)
            End If
            Range("A2").FormulaR1C1 = "=ISBLANK(R1C1)"
            If (Range("A2").Value) Then ' wrong because no answer entered
                StudMarks = 0
                WrongF = 1
                TickCross = "û" ' wingdings cross
                GARcol = 3 ' red
                ' make this answer an Excel #VALUE! error so consistent errors don't pick
up 0
                Sheets("SOLUTIONS").Select
                Range(Cells(xx, 17), Cells(xx, 17)).Value = "#VALUE!"
                Sheets("MASTER").Select
            Else ' genuine answer, so check to see if correct
                If (IsNumeric(Answer(WhichAns))) Then ' numeric answer
                    If (IsNumeric(StudSols(WhichAns))) Then ' right form
                        ' check within tolerance

                        If (Abs((StudSols(WhichAns) - Answer(WhichAns))) <=
Tolerance(WhichAns)) Then ' CORRECT! (within tolerance)
                            ' need to force a consistent error check?
                            If (ForceCE(WhichAns) = 0) Then ' definitely OK
                                StudMarks = Marks(xx - 4)
                                GARcol = 50 ' green
                                TickCross = "ü" ' wingdings tick
                            Else ' must satisfy the consistent error check to be OK
                                ' obtain the answer that the student must have to be
consistent
                                Sheets("SOLUTIONS").Select
                                If (Not (VarType(Range(Cells(xx, 11), Cells(xx,
11)).Value) = vbError)) Then ' only do this if it's a real value
                                    CESol = Range(Cells(xx, 11), Cells(xx, 11)).Value
                                    If (IsNumeric(CESol)) Then ' numeric answer
                                        If (Abs((StudSols(WhichAns) - CESol)) >
Tolerance(WhichAns)) Then ' fails consistent error check
                                            StudMarks = 0
                                            TickCross = "û" ' wingdings cross
                                            FeedBackText = "Although it matches the
solutions, your answer is NOT consistent with earlier errors, and so is wrong. "
                                            GARcol = 45 ' amber
                                        Else 'OK
```

```
                                        StudMarks = Marks(xx - 4)
                                        GARcol = 50 ' green
                                        TickCross = "ü" ' wingdings tick
                                        WrongF = 0
                                    End If ' else OK
                                End If
                            End If
                            Sheets("MASTER").Select
                        End If
                    Else ' answer initially wrong - do further checks
                        StudMarks = 0
                        WrongF = 1
                        TickCross = "û" ' wingdings cross
                        GARcol = 3 ' red

                        ' Alternative decimal place check first

                        If ((AltDecCheck = 1) And (Abs(Int(StudSols(WhichAns) * (10 ^
AltDec)) / (10# ^ AltDec) - StudSols(WhichAns)) <= 0.00000000001) And
(Abs(StudSols(WhichAns) - Round(Answer(WhichAns), AltDec)) <= Tolerance(WhichAns))) Then
                            ' It is of this number dps and it's OK
                            StudMarks = Marks(xx - 4)
                            GARcol = 4 ' lt green colour
                            WrongF = 0
                            TickCross = "ü" ' wingdings tick
                        Else ' check for errors now
                            If (CommError(WhichAns)) Then '
                                ' common error checks, plus for validity, first
                                Sheets("ERROR").Select
                                NumComm = Range(Cells(CommRow(WhichAns), 2),
Cells(CommRow(WhichAns), 2)).Value
                                For vv = 1 To NumComm
                                    If (Not (VarType(Range(Cells(CommRow(WhichAns), 3
* vv), Cells(CommRow(WhichAns), 3 * vv)).Value)) = vbError) Then ' only do this if it's a
real value
                                        CESol = Range(Cells(CommRow(WhichAns), 3 *
vv), Cells(CommRow(WhichAns), 3 * vv)).Value
                                        If (IsNumeric(CESol)) Then ' numeric answer
check (given student previous solutions)
                                            If (CESol = 999) Then ' validity check
for max
                                                ValToCheck =
Range(Cells(CommRow(WhichAns), 3 * vv + 1), Cells(CommRow(WhichAns), 3 * vv + 1)).Value
                                                If (StudSols(WhichAns) > ValToCheck)
Then ' greater than max
                                                    FeedBackText = "Your answer is
greater than the possible maximum value of " & ValToCheck & ". "
                                                    If (CoErrNY) Then
                                                        GARcol = 39 ' purple
                                                    Else
                                                        GARcol = 54 ' plum
                                                    End If
                                                    ValidYN = 0 ' not valid!
                                                    StudMarks =
Range(Cells(CommRow(WhichAns), 3 * vv + 2), Cells(CommRow(WhichAns), 3 * vv + 2)).Value
                                                End If
                                            ElseIf (CESol = -999) Then ' validity
check for min
                                                ValToCheck =
Range(Cells(CommRow(WhichAns), 3 * vv + 1), Cells(CommRow(WhichAns), 3 * vv + 1)).Value
                                                If (StudSols(WhichAns) < ValToCheck)
Then ' less than min
                                                    FeedBackText = "Your answer is
less than the possible minimum value of " & ValToCheck & ". "
                                                    If (CoErrNY) Then
                                                        GARcol = 39 ' purple
                                                    Else
                                                        GARcol = 54 ' plum
                                                    End If
                                                    ValidYN = 0 ' not valid!
```

```
                                                        StudMarks =
Range(Cells(CommRow(WhichAns), 3 * vv + 2), Cells(CommRow(WhichAns), 3 * vv + 2)).Value
                                                    End If
                                                ElseIf (CoErrNY) Then ' not a validity
check, but also common error not found already
                                                    If (Abs((StudSols(WhichAns) - CESol))
<= Tolerance(WhichAns)) Then ' satisfies common error check
                                                        FeedBackText = FeedBackText &
Range(Cells(CommRow(WhichAns), 3 * vv + 1), Cells(CommRow(WhichAns), 3 * vv + 1)).Value &
" "
                                                        If (ValidYN) Then
                                                            GARcol = 7 ' pink
                                                        Else
                                                            GARcol = 54 ' plum
                                                        End If
                                                        CoErrNY = 0 ' common error found
                                                        ' do not overwrite validity check
error marks - they take precedence
                                                        If (ValidYN) Then StudMarks =
Range(Cells(CommRow(WhichAns), 3 * vv + 2), Cells(CommRow(WhichAns), 3 * vv + 2)).Value
                                                    End If
                                                End If
                                            End If
                                        End If
                                    Next 'vv
                                    Sheets("MASTER").Select
                                End If

                                If (ValidYN And CoErrNY And ConsError(WhichAns)) Then
                                    ' consistent error check (only do if validity check
not failed!)
                                    Sheets("SOLUTIONS").Select

                                    If (Not (VarType(Range(Cells(xx, 11), Cells(xx,
11)).Value) = vbError)) Then ' only do if it's a real value
                                        CESol = Range(Cells(xx, 11), Cells(xx, 11)).Value
                                        If (IsNumeric(CESol)) Then ' numeric answer given
student previous solutions
                                            If (Abs((StudSols(WhichAns) - CESol)) <=
Tolerance(WhichAns)) Then ' passes consistent error check
                                                StudMarks = MarksCE(WhichAns)
                                                GARcol = 5 ' blue
                                                FeedBackText = FeedBackText & "Answer is
incorrect, but it is consistent with other errors. "
                                            End If
                                        End If
                                    End If
                                    Sheets("MASTER").Select
                                End If
                            End If
                        End If
                    Else ' student entered non-numeric answer, which is wrong
                        StudMarks = 0
                        WrongF = 1
                        TickCross = "û" ' wingdings cross
                        GARcol = 3 ' red
                    End If
                Else ' text solution to check
                    If (StudSols(WhichAns) = Answer(WhichAns)) Then ' CORRECT!
                        ' need to force a consistent error check?
                        If (ForceCE(WhichAns) = 0) Then ' definitely OK
                            StudMarks = Marks(xx - 4)
                            GARcol = 50 ' green colour
                            TickCross = "ü" ' wingdings tick
                            WrongF = 0
                        Else ' must satisfy the consistent error check to be OK
                            ' obtain the answer that the student must have to be
consistent

                            Sheets("SOLUTIONS").Select
```

```
                            If (Not (VarType(Range(Cells(xx, 11), Cells(xx, 11)).Value))
= vbError) Then ' only do if it's a real value
                                CESol = Range(Cells(xx, 11), Cells(xx, 11)).Value
                                If (Not (StudSols(WhichAns) = CESol)) Then ' fails
consistent error check
                                    StudMarks = 0
                                    TickCross = "û" ' wingdings cross
                                    FeedBackText = "Although it matches the solutions,
your answer is NOT consistent with earlier errors, and so is wrong. "
                                    GARcol = 45 ' amber
                                Else 'OK
                                    StudMarks = Marks(xx - 4)
                                    GARcol = 50 ' green
                                    TickCross = "ü" ' wingdings tick
                                    WrongF = 0
                                End If ' else OK
                            End If
                            Sheets("MASTER").Select
                        End If
                    Else ' answer initially wrong - do further checks
                        StudMarks = 0
                        WrongF = 1
                        TickCross = "û" ' wingdings cross
                        GARcol = 3 ' red

                        If (CommError(WhichAns)) Then
                            ' common error checks first
                            Sheets("ERROR").Select
                            NumComm = Range(Cells(CommRow(WhichAns), 2),
Cells(CommRow(WhichAns), 2)).Value

                            For vv = 1 To NumComm
                                If (Not (VarType(Range(Cells(CommRow(WhichAns), 3 * vv),
Cells(CommRow(WhichAns), 3 * vv)).Value)) = vbError) Then ' only do if it's a real value
                                    CESol = Range(Cells(CommRow(WhichAns), 3 * vv),
Cells(CommRow(WhichAns), 3 * vv)).Value
                                    If (StudSols(WhichAns) = CESol) Then ' satisfies
common error check
                                        StudMarks = Range(Cells(CommRow(WhichAns), 3 * vv
+ 2), Cells(CommRow(WhichAns), 3 * vv + 2)).Value
                                        FeedBackText = FeedBackText &
Range(Cells(CommRow(WhichAns), 3 * vv + 1), Cells(CommRow(WhichAns), 3 * vv + 1)).Value &
" "
                                        GARcol = 7 ' pink
                                    End If
                                End If
                            Next 'vv
                            Sheets("MASTER").Select
                        End If

                        If (ConsError(WhichAns)) Then
                            ' consistent error check
                            Sheets("SOLUTIONS").Select

                            If (Not (VarType(Range(Cells(xx, 11), Cells(xx, 11)).Value))
= vbError) Then ' only do if it's a real value
                                CESol = Range(Cells(xx, 11), Cells(xx, 11)).Value
                                If (StudSols(WhichAns) = CESol) Then ' passes consistent
error check
                                    StudMarks = MarksCE(WhichAns)
                                    GARcol = 5 ' blue
                                    FeedBackText = FeedBackText & "Answer is incorrect,
but it is consistent with other errors. "
                                End If
                            End If
                            Sheets("MASTER").Select
                        End If
                    End If
                End If
```

```
            End If

            If (WrongF) Then
                FeedBackText = FeedBackText & "The correct answer is ... " &
Answer(WhichAns) & ". "
            End If

            ' add apostrophe to text, so excel will not hit problem with answer starting
with =
            Range(Cells(ThisRow, WhichCol), Cells(ThisRow, WhichCol)).Select
            If (Not (IsNumeric(Answer(WhichAns)))) Then
                Selection.Value = "'" & Answer(WhichAns)
            Else
                Selection.Value = Answer(WhichAns)
            End If

            Range(Cells(ThisRow + 1, WhichCol), Cells(ThisRow + 1, WhichCol)).Select
            If (Not (IsNumeric(StudSols(WhichAns)))) Then
                Selection.Value = "'" & StudSols(WhichAns)
            Else
                Selection.Value = StudSols(WhichAns)
            End If
            Selection.Font.ColorIndex = GARcol
            Range(Cells(ThisRow + 2, WhichCol), Cells(ThisRow + 2, WhichCol)).Select
            Selection.Value = TickCross
            Selection.Font.Name = "Wingdings"
            Range(Cells(ThisRow + 3, WhichCol), Cells(ThisRow + 3, WhichCol)).Value =
StudMarks
            Range(Cells(ThisRow + 4, WhichCol), Cells(ThisRow + 4, WhichCol)).Value =
FeedBackText

            ' now apply formatting to this block

            Range(Cells(ThisRow, WhichCol), Cells(ThisRow + 4, WhichCol)).Select
            With Selection.Borders(xlEdgeLeft)
                .LineStyle = xlContinuous
                .Weight = xlThin
                .ColorIndex = xlAutomatic
            End With
            With Selection.Borders(xlEdgeTop)
                .LineStyle = xlContinuous
                .Weight = xlThin
                .ColorIndex = xlAutomatic
            End With
            With Selection.Borders(xlEdgeBottom)
                .LineStyle = xlContinuous
                .Weight = xlThin
                .ColorIndex = xlAutomatic
            End With
            With Selection.Borders(xlEdgeRight)
                .LineStyle = xlContinuous
                .Weight = xlThin
                .ColorIndex = xlAutomatic
            End With

            WhichAns = WhichAns + 1
            ThisRow = ThisRow + 5
            QRunSum = QRunSum + StudMarks
        ElseIf (QOrTotal(xx - 4) < 0) Then ' graph - set format, marks to be added
manually later
            Range(Cells(ThisRow, WhichCol), Cells(ThisRow + 1, WhichCol)).Value = "GRAPH"
            Range(Cells(ThisRow, WhichCol), Cells(ThisRow + 4, WhichCol)).Select
            With Selection.Borders(xlEdgeLeft)
                .LineStyle = xlContinuous
                .Weight = xlThin
                .ColorIndex = xlAutomatic
            End With
            With Selection.Borders(xlEdgeTop)
                .LineStyle = xlContinuous
                .Weight = xlThin
```

```
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeBottom)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        With Selection.Borders(xlEdgeRight)
            .LineStyle = xlContinuous
            .Weight = xlThin
            .ColorIndex = xlAutomatic
        End With
        WhichAns = WhichAns + 1
        ThisRow = ThisRow + 5
    Else ' Q total
        Range(Cells(ThisRow, WhichCol), Cells(ThisRow, WhichCol)).Select
        Selection.Value = QRunSum
        With Selection.Interior
            .ColorIndex = 36
            .Pattern = xlSolid
        End With

        ThisRow = ThisRow + 1
        NewTotal = NewTotal + QRunSum
        QRunSum = 0 ' new (part) question starts
    End If
Next ' xx (question)

' now print overall total

Range(Cells(ThisRow, WhichCol), Cells(ThisRow, WhichCol)).Select
Selection.Value = NewTotal
With Selection.Interior
    .ColorIndex = 6
    .Pattern = xlSolid
End With

' now print generic feedback for disability flag

Range(Cells(ThisRow + 1, WhichCol), Cells(ThisRow + 1, WhichCol)).Value = Feed1

' now apply formatting to the feedback block

Range(Cells(ThisRow + 1, WhichCol), Cells(ThisRow + 2, WhichCol)).Select
With Selection.Borders(xlEdgeLeft)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeTop)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
With Selection.Borders(xlEdgeRight)
    .LineStyle = xlContinuous
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With

' now copy marks, feedback etc onto the student's sheet
' first read everything into arrays from MASTER sheet

ThisRow = 5
WhichQ = 0
```

```
    QRunSum = 0 ' for calculating new part-Q total
    NewTotal = 0 ' for calculating new overall total
    WhichAns = 0

    Sheets("MASTER").Select
    Range("A1").Clear
    Range("A2").Clear

    While (ThisRow < TotalRow)
        RowType = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value
        If (RowType = 0) Then ' feedback info to copy
            WhichAns = WhichAns + 1
            FeedBack(WhichAns, 1) = Range(Cells(ThisRow + 2, WhichCol), Cells(ThisRow +
2, WhichCol)).Value ' tick/cross
            FeedBack(WhichAns, 2) = Range(Cells(ThisRow + 4, WhichCol), Cells(ThisRow +
4, WhichCol)).Value ' actual feedback (may be blank)
            QRunSum = QRunSum + Range(Cells(ThisRow + 3, WhichCol), Cells(ThisRow + 3,
WhichCol)).Value
            ThisRow = ThisRow + 5 ' assumes all blocks of info are in 5s
        Else ' Q total
            WhichQ = WhichQ + 1
            QTotal(WhichQ) = QRunSum
            Range(Cells(ThisRow, WhichCol), Cells(ThisRow, WhichCol)).Value = QRunSum
            ThisRow = ThisRow + 1
            NewTotal = NewTotal + QRunSum
            QRunSum = 0 ' new (part) question starts
         End If
    Wend
    Range(Cells(TotalRow, WhichCol), Cells(TotalRow, WhichCol)).Value = NewTotal

    ' now copy to student sheet
    Worksheets(ii).Activate

    ' add following info onto student sheet, but only for Word feedback version
    If (WhichType = vbYes) Then
        ActiveSheet.Range("B2").Value = MaxCol + 2
        ActiveSheet.Range("B5").Value = WhichQ
    End If

    ' add everything else

    NumFeed = 0
    For j = 1 To WhichAns
        ' Tick marks and feedback
        Range(Cells(RowsColsM(j, 1), RowsColsM(j, 2)), Cells(RowsColsM(j, 1),
RowsColsM(j, 2))).Select
        Selection.Value = FeedBack(j, 1)
        With Selection.Font
            .Name = "Wingdings"
        End With
        If (Not (IsNumeric(FeedBack(j, 2)))) Then ' assume it's not empty
            NumFeed = NumFeed + 1
            Range(Cells(RowsColsF(j, 1) - 1, RowsColsF(j, 2)), Cells(RowsColsF(j, 1) - 1,
RowsColsF(j, 2))).Value = "[" & NumFeed & "]"
        Else
            Range(Cells(RowsColsF(j, 1) - 1, RowsColsF(j, 2)), Cells(RowsColsF(j, 1) - 1,
RowsColsF(j, 2))).Value = ""
        End If
        Range(Cells(RowsColsF(j, 1), RowsColsF(j, 2)), Cells(RowsColsF(j, 1),
RowsColsF(j, 2))).Value = FeedBack(j, 2)
    Next 'j

    ' Q totals and overall total

    For j = 1 To WhichQ
        Range(Cells(RowsColsT(j, 1), RowsColsT(j, 2)), Cells(RowsColsT(j, 1),
RowsColsT(j, 2))).Value = QTotal(j)
    Next 'j
    Range(Cells(RowsColsT(WhichQ + 1, 1), RowsColsT(WhichQ + 1, 2)),
Cells(RowsColsT(WhichQ + 1, 1), RowsColsT(WhichQ + 1, 2))).Value = NewTotal
```

```
    Range(Cells(FeedRC1(1), FeedRC1(2)), Cells(FeedRC1(1), FeedRC1(2))).Value = Feed1
Next 'ii (student worksheet)

Sheets("MASTER").Select

Range("D2").Select
ActiveWindow.WindowState = xlMaximized

End Sub
```

Sub ChangeMarks()

```
'
' ChangeMarks Macro
' Macro written Sep 2008 by Karen Ayres
'
' Amends marks etc. on student sheets after amendments have been manually
' entered into MASTER sheet

ActiveWindow.WindowState = xlMinimized
Worksheets("MASTER").Select
TotalRow = Range("D4").Value

WhichType = MsgBox("Is this assignment using Word forms for solutions and feedback?",
vbYesNo)

NumSheets = Sheets.Count

'move non-student sheets to be last sheets if they aren't already

Sheets("MASTER").Select
Sheets("MASTER").Move after:=Sheets(NumSheets)

Sheets("ERROR").Select
Sheets("ERROR").Move after:=Sheets(NumSheets)

Sheets("SOLUTIONS").Select
Sheets("SOLUTIONS").Move after:=Sheets(NumSheets)

Sheets("DATA").Select
Sheets("DATA").Move after:=Sheets(NumSheets)

Sheets("SUMMARY").Select
Sheets("SUMMARY").Move after:=Sheets(NumSheets)

Sheets("STUDENTS").Select
Sheets("STUDENTS").Move after:=Sheets(NumSheets)

NumSheets = NumSheets - 6 ' number of student solution sheets

' first read in the locations to print things to sheets

Sheets("SOLUTIONS").Select
FinalRow = Range("D1").Value
WhichQ = 0
WhichAns = 1
For i = 5 To FinalRow
    If (Range(Cells(i, 4), Cells(i, 4)).Value = 0) Then
        ' total for (part)question
        WhichQ = WhichQ + 1
        If (WhichType = vbYes) Then
            RowsColsT(WhichQ, 1) = 5
            RowsColsT(WhichQ, 2) = 2 + WhichQ
        Else
            RowsColsT(WhichQ, 1) = Range(Cells(i, 6), Cells(i, 6)).Value
            RowsColsT(WhichQ, 2) = Range(Cells(i, 7), Cells(i, 7)).Value
        End If
    Else
        Marks(WhichAns) = Range(Cells(i, 2), Cells(i, 2)).Value
        If (WhichType = vbYes) Then
```

```
            RowsColsM(WhichAns, 1) = 2
            RowsColsF(WhichAns, 1) = 4
        Else
            RowsColsM(WhichAns, 1) = Range(Cells(i, 6), Cells(i, 6)).Value
            RowsColsF(WhichAns, 1) = Range(Cells(i, 8), Cells(i, 8)).Value
        End If
        RowsColsM(WhichAns, 2) = Range(Cells(i, 7), Cells(i, 7)).Value
        RowsColsF(WhichAns, 2) = Range(Cells(i, 9), Cells(i, 9)).Value
        WhichAns = WhichAns + 1
    End If
Next 'i

WhichQ = WhichQ + 1 ' overall total now
If (WhichType = vbYes) Then
    RowsColsT(WhichQ, 1) = 5
    RowsColsT(WhichQ, 2) = 2 + WhichQ
    FeedRC1(1) = 6
    FeedRC1(2) = 3
    FeedRC2(1) = 6
    FeedRC2(2) = 4
Else
    RowsColsT(WhichQ, 1) = Range("L1").Value
    RowsColsT(WhichQ, 2) = Range("N1").Value
    FeedRC1(1) = Range("L2").Value
    FeedRC1(2) = 1
    FeedRC2(1) = Range("N2").Value
    FeedRC2(2) = 1
End If

' copy all of what's on MASTER sheet to student sheet (except solutions)

For i = 1 To NumSheets
    Worksheets(i).Activate
    WhichCol = 5 + Range("A5000").Value
    ThisRow = 5
    WhichQ = 0
    QRunSum = 0 ' for calculating new part-Q total
    NewTotal = 0 ' for calculating new overall total
    WhichAns = 0

    ' first read everything into arrays from MASTER sheet

    Sheets("MASTER").Select
    While (ThisRow < TotalRow)
        RowType = Range(Cells(ThisRow, 1), Cells(ThisRow, 1)).Value
        If (RowType = 0) Then ' feedback info to copy
            WhichAns = WhichAns + 1
            If (Range(Cells(ThisRow + 3, WhichCol), Cells(ThisRow + 3, WhichCol)).Value >
Marks(WhichAns)) Then ' error in mark allocation
                xxx = MsgBox("Error - marks awarded for Q. " & Range(Cells(ThisRow, 2),
Cells(ThisRow, 2)).Value & " for student " & Range(Cells(4, WhichCol), Cells(4,
WhichCol)).Value & " exceed number allocated to question. Click OK to ignore and proceed,
or Cancel to manually fix.", vbOKCancel)
                If (xxx = vbCancel) Then GoTo JumpToEnd
            End If
            FeedBack(WhichAns, 1) = Range(Cells(ThisRow + 2, WhichCol), Cells(ThisRow +
2, WhichCol)).Value ' tick/cross
            FeedBack(WhichAns, 2) = Range(Cells(ThisRow + 4, WhichCol), Cells(ThisRow +
4, WhichCol)).Value ' actual feedback (may be blank)
            QRunSum = QRunSum + Range(Cells(ThisRow + 3, WhichCol), Cells(ThisRow + 3,
WhichCol)).Value
            ThisRow = ThisRow + 5 ' assumes all blocks of info are in 5s
        Else ' Q total
            WhichQ = WhichQ + 1
            QTotal(WhichQ) = QRunSum
            Range(Cells(ThisRow, WhichCol), Cells(ThisRow, WhichCol)).Value = QRunSum
            ThisRow = ThisRow + 1
            NewTotal = NewTotal + QRunSum
            QRunSum = 0 ' new (part) question starts
         End If
```

```
    Wend
    Range(Cells(TotalRow, WhichCol), Cells(TotalRow, WhichCol)).Value = NewTotal
    Feed1 = Range(Cells(TotalRow + 1, WhichCol), Cells(TotalRow + 1, WhichCol)).Value
    Feed2 = Range(Cells(TotalRow + 2, WhichCol), Cells(TotalRow + 2, WhichCol)).Value

    ' now copy to student sheet
    Worksheets(i).Activate

    NumFeed = 0
    For j = 1 To WhichAns
        ' Tick marks and feedback

        Range(Cells(RowsColsM(j, 1), RowsColsM(j, 2)), Cells(RowsColsM(j, 1),
RowsColsM(j, 2))).Value = FeedBack(j, 1)
        If (Not (IsNumeric(FeedBack(j, 2)))) Then ' assume it's not empty
            NumFeed = NumFeed + 1
            Range(Cells(RowsColsF(j, 1) - 1, RowsColsF(j, 2)), Cells(RowsColsF(j, 1) - 1,
RowsColsF(j, 2))).Value = "[" & NumFeed & "]"
        Else
            Range(Cells(RowsColsF(j, 1) - 1, RowsColsF(j, 2)), Cells(RowsColsF(j, 1) - 1,
RowsColsF(j, 2))).Value = ""
        End If
        Range(Cells(RowsColsF(j, 1), RowsColsF(j, 2)), Cells(RowsColsF(j, 1),
RowsColsF(j, 2))).Value = FeedBack(j, 2)
    Next 'j

    ' Q totals and overall total

    For j = 1 To WhichQ
        Range(Cells(RowsColsT(j, 1), RowsColsT(j, 2)), Cells(RowsColsT(j, 1),
RowsColsT(j, 2))).Value = QTotal(j)
    Next 'j
    Range(Cells(RowsColsT(WhichQ + 1, 1), RowsColsT(WhichQ + 1, 2)),
Cells(RowsColsT(WhichQ + 1, 1), RowsColsT(WhichQ + 1, 2))).Value = NewTotal
    Range(Cells(FeedRC1(1), FeedRC1(2)), Cells(FeedRC1(1), FeedRC1(2))).Value = Feed1
    Range(Cells(FeedRC2(1), FeedRC2(2)), Cells(FeedRC2(1), FeedRC2(2))).Value = Feed2
Next 'i

JumpToEnd:

Worksheets("SUMMARY").Select
Range("A1").Select
Worksheets("STUDENTS").Select
Range("A1").Select
Worksheets("DATA").Select
Range("A1").Select
Worksheets("SOLUTIONS").Select
Range("A1").Select
Worksheets("ERROR").Select
Range("A1").Select
Worksheets("MASTER").Select
Range("H2").Select

ActiveWindow.WindowState = xlMaximized

End Sub
```

Sub CreateMergeTable()

```
'
' OpenAllExcel Macro
' Macro written Sept. 2008 by Karen Ayres
'
' Macro reads solutions, feedback, etc from student sheets and prints all into
' an Excel file that can be used in a mail merge to create the Word formatted
' feedback sheets

ActiveWindow.WindowState = xlMinimized

NumSheets = Sheets.Count
```

```
'move non-student sheets to be last sheets if they aren't already

Sheets("MASTER").Select
AllMarks = Range("C4").Value
Sheets("MASTER").Move after:=Sheets(NumSheets)

Sheets("ERROR").Select
Sheets("ERROR").Move after:=Sheets(NumSheets)

Sheets("SOLUTIONS").Select
Sheets("SOLUTIONS").Move after:=Sheets(NumSheets)

Sheets("DATA").Select
Sheets("DATA").Move after:=Sheets(NumSheets)

Sheets("SUMMARY").Select
Sheets("SUMMARY").Move after:=Sheets(NumSheets)

Sheets("STUDENTS").Select
Sheets("STUDENTS").Move after:=Sheets(NumSheets)

Set MergeSheet = Worksheets.Add
MergeSheet.Move after:=Sheets(NumSheets + 1)
MyFileName = ActiveWorkbook.Path & "/merge.xlsx"

For i = 1 To NumSheets - 6 ' one sheet per student
    NumFeed = 0
    Worksheets(i).Activate
    StudName = ActiveSheet.Range("A1").Value
    StudID = ActiveSheet.Range("B1").Value
    LastCol = ActiveSheet.Range("B2").Value - 2
    NumMark = ActiveSheet.Range("B5").Value ' number of stop-off mark totals
    MergeSheet.Activate
    ActiveSheet.Range(Cells(i + 1, 1), Cells(i + 1, 1)).Value = StudName
    ActiveSheet.Range(Cells(i + 1, 2), Cells(i + 1, 2)).Value = StudID
    For j = 1 To LastCol
        Worksheets(i).Activate
        If (IsNumeric(ActiveSheet.Range(Cells(1, j + 2), Cells(1, j + 2)).Value)) Then
            Range("A5001").Value = ActiveSheet.Range(Cells(1, j + 2), Cells(1, j +
2)).Value
        Else
            Range("A5001").Value = "'" & ActiveSheet.Range(Cells(1, j + 2), Cells(1, j +
2)).Value
        End If
        Range("A5002").FormulaR1C1 = "=ISBLANK(R5001C1)"
        If (Range("A5002").Value) Then ' no answer entered
            StudAns = "____"
        Else
            StudAns = Range("A5001").Value
        End If
        StudTick = ActiveSheet.Range(Cells(2, j + 2), Cells(2, j + 2)).Value
        StudFeed = ActiveSheet.Range(Cells(3, j + 2), Cells(3, j + 2)).Value
        MergeSheet.Activate
        If (IsNumeric(StudAns)) Then
            ActiveSheet.Range(Cells(i + 1, 3 * (j - 1) + 3), Cells(i + 1, 3 * (j - 1) +
3)).Value = StudAns
        Else
            ActiveSheet.Range(Cells(i + 1, 3 * (j - 1) + 3), Cells(i + 1, 3 * (j - 1) +
3)).Value = "'" & StudAns
        End If
        ActiveSheet.Range(Cells(i + 1, 3 * (j - 1) + 4), Cells(i + 1, 3 * (j - 1) +
4)).Value = StudTick
        ActiveSheet.Range(Cells(i + 1, 3 * (j - 1) + 5), Cells(i + 1, 3 * (j - 1) +
5)).Value = StudFeed
    Next 'j
    WhichCol = 2 + (LastCol * 3) + 1
    Range("A5001").Clear
    Range("A5002").Clear
```

```
    For j = 1 To LastCol
        Worksheets(i).Activate
        If (Not (IsNumeric(ActiveSheet.Range(Cells(4, j + 2), Cells(4, j + 2)).Value)))
Then
            StudFeedB = ActiveSheet.Range(Cells(4, j + 2), Cells(4, j + 2)).Value
            NumFeed = NumFeed + 1
            MergeSheet.Activate
            ActiveSheet.Range(Cells(i + 1, WhichCol), Cells(i + 1, WhichCol)).Value =
ActiveSheet.Range(Cells(i + 1, 3 * (j - 1) + 5), Cells(i + 1, 3 * (j - 1) + 5)).Value & "
" & StudFeedB & Chr(13)
        End If
        WhichCol = WhichCol + 1
    Next 'j

    Worksheets(i).Activate

    Feed1 = ActiveSheet.Range("C6").Value
    Feed2 = ActiveSheet.Range("D6").Value

    If (Not (IsNumeric(Feed1))) Then
        Feed1 = Chr(13) & Feed1
    End If
    If (Not (IsNumeric(Feed2))) Then
        Feed2 = Chr(13) & Feed2
    End If

    MergeSheet.Activate
    ActiveSheet.Range(Cells(i + 1, WhichCol), Cells(i + 1, WhichCol)).Value = Feed1
    ActiveSheet.Range(Cells(i + 1, WhichCol + 1), Cells(i + 1, WhichCol + 1)).Value =
Feed2
    WhichCol = WhichCol + 2

    Worksheets(i).Activate
    For j = 1 To NumMark
        Marks(j) = ActiveSheet.Range(Cells(5, j + 2), Cells(5, j + 2)).Value
    Next 'j
    TotalMark = ActiveSheet.Range(Cells(5, NumMark + 3), Cells(5, NumMark + 3)).Value
    TotalPct = TotalMark / AllMarks
    MergeSheet.Activate
    For j = 1 To NumMark
        ActiveSheet.Range(Cells(i + 1, WhichCol), Cells(i + 1, WhichCol)).Value =
Marks(j)
        WhichCol = WhichCol + 1
    Next 'j
    ActiveSheet.Range(Cells(i + 1, WhichCol), Cells(i + 1, WhichCol)).Value = TotalMark
    ActiveSheet.Range(Cells(i + 1, WhichCol + 1), Cells(i + 1, WhichCol + 1)).Value =
Round(TotalPct * 100, 2)
Next 'i

' Now add headings (field names) to table
MergeSheet.Activate
ActiveSheet.Range("A1").Value = "StudName"
ActiveSheet.Range("B1").Value = "StudNum"

For j = 1 To LastCol
    ActiveSheet.Range(Cells(1, 3 * (j - 1) + 3), Cells(1, 3 * (j - 1) + 3)).Value = "A" &
j
    ActiveSheet.Range(Cells(1, 3 * (j - 1) + 4), Cells(1, 3 * (j - 1) + 4)).Value = "M" &
j
    ActiveSheet.Range(Cells(1, 3 * (j - 1) + 5), Cells(1, 3 * (j - 1) + 5)).Value = "F" &
j
Next 'j
WhichCol = 2 + (LastCol * 3)
For j = 1 To LastCol
    ActiveSheet.Range(Cells(1, WhichCol + j), Cells(1, WhichCol + j)).Value = "Feed" & j
Next 'j
WhichCol = WhichCol + LastCol
ActiveSheet.Range(Cells(1, WhichCol + 1), Cells(1, WhichCol + 1)).Value = "Feed" &
(LastCol + 1)
```

```
ActiveSheet.Range(Cells(1, WhichCol + 2), Cells(1, WhichCol + 2)).Value = "Feed" &
(LastCol + 2)

WhichCol = WhichCol + 2
For j = 1 To NumMark
    ActiveSheet.Range(Cells(1, WhichCol + j), Cells(1, WhichCol + j)).Value = "Mark" & j
Next 'j
ActiveSheet.Range(Cells(1, WhichCol + NumMark + 1), Cells(1, WhichCol + NumMark +
1)).Value = "Total"
ActiveSheet.Range(Cells(1, WhichCol + NumMark + 2), Cells(1, WhichCol + NumMark +
2)).Value = "Percent"

' must be saved in text format
Cells.Select
Selection.NumberFormat = "@"
Range("A1").Select

Application.DisplayAlerts = False
MergeSheet.Move
ActiveSheet.Select
ActiveSheet.Name = "Sheet1"
ActiveWorkbook.SaveAs Filename:=MyFileName
ActiveWorkbook.Close
Application.DisplayAlerts = True

Worksheets("SUMMARY").Select
Range("A1").Select
Worksheets("STUDENTS").Select
Range("A1").Select
Worksheets("DATA").Select
Range("A1").Select
Worksheets("SOLUTIONS").Select
Range("A1").Select
Worksheets("ERROR").Select
Range("A1").Select
Worksheets("MASTER").Select
Range("H2").Select

ActiveWindow.WindowState = xlMaximized

End Sub
Sub PrintMarks()
'
' PrintMarks Macro
' Macro written Sep 2008 by Karen Ayres
'
' Creates a summary sheet with names, student numbers, and overall mark and percentage
'

ActiveWindow.WindowState = xlMinimized
Worksheets("MASTER").Select
TotalMark = Range("C4").Value
TotalRow = Range("D4").Value

Worksheets("STUDENTS").Select
Cells(2, 1).Select
Selection.End(xlDown).Select
NumStudents = Selection.Row - 1

Range(Cells(2, 1), Cells(NumStudents + 1, 3)).Select
Selection.Copy
Worksheets("SUMMARY").Select
Range("A3").Select
ActiveSheet.Paste

Range("D2").Value = "Mark (/" & TotalMark & ")"

Worksheets("MASTER").Select

For i = 1 To NumStudents
```

```
    ' copy totalmark into array first
    StudTotal(i) = Range(Cells(TotalRow, 5 + i), Cells(TotalRow, 5 + i)).Value
Next 'i

Worksheets("SUMMARY").Select

For i = 1 To NumStudents
    Range(Cells(i + 2, 4), Cells(i + 2, 4)).Value = StudTotal(i)
    Range(Cells(i + 2, 5), Cells(i + 2, 5)).Value = StudTotal(i) / TotalMark
Next 'i

Columns("A:E").Select
With Selection
   .HorizontalAlignment = xlCenter
End With

Columns("E:E").NumberFormat = "0.00%"

Range("A1").Select
ActiveWindow.WindowState = xlMaximized
End Sub
```

Appendix C – Master code

The code for the Word Master.doc document, which opens all Word student files saved in a folder called Forms, and saves the form entries in text files in a folder called Files, is given below.

```
Sub SaveAllForms()
'
' SaveAllForms Macro
' Macro written August 2008 by Karen Ayres
'

Application.DisplayAlerts = False
MsgBox "Note: this function will NOT open files that are READ-ONLY. If you have any files
like this, click Cancel and change their properties before running this macro",
vbOKCancel

ActiveDocument.ActiveWindow.WindowState = wdWindowStateMinimize
MyFilePath = ActiveDocument.Path & "\Forms\"
MySavePath = ActiveDocument.Path & "\Files\"

MyFileToOpen = Dir(MyFilePath & "*.doc?")

If MyFileToOpen = "" Then MsgBox "There were no files found - make sure they all have
.doc? extension and are in subfolder named Forms."

FileNum = 0

Do While MyFileToOpen <> ""
    FileNum = FileNum + 1
    MyFileName = MyFilePath & MyFileToOpen

    Documents.Open FileName:=MyFileName, ReadOnly:=False, _
        Format:=wdOpenFormatAuto

    SaveFileName = MySavePath & "form" & FileNum & ".txt"
    ActiveDocument.SaveAs FileName:=SaveFileName, FileFormat:=wdFormatText, _
         SaveNativePictureFormat:=False, SaveFormsData:=True

    ActiveDocument.Close
    MyFileToOpen = Dir
Loop
ActiveDocument.ActiveWindow.WindowState = wdWindowStateMaximize
Application.DisplayAlerts = True
End Sub
```