# UG Exam Performance Feedback

## Third Year - Semester 1

**COMP 31111**  
**Verified Development**  
**Richard Banach**

**Comments:**  
Q.1  reasonably well answered on the whole  
Q.2  with one exception, rather poorly answered  
Q.3  answers could have been better, but not too bad on the whole

---

**COMP 32111**  
**System-on-Chip modelling with System C**  
**Steve Furber**  
**Alasdair Rawsthorne**

**Comments:**  
It was clear from the lab that the cohort taking this course is strong, and the exam marks reflect this strength, with the overall marks ranging from 63% to 83%.

Q1 (compulsory, 10-part question): All students did OK on this, though parts c) & d) showed up a general inability to do estimated arithmetic with big numbers.  
AR: anything more?  
AR: Again, the answers to my parts showed some difficulty in handling times, clock rates and bandwidths. Disappointing, given the good quality of the cohort.

Q2: No attempts, which is a bit disappointing!

Q3: 4 students attempted this question, all obtaining reasonable marks. Weaker answers tended to be a bit woolly, and in part e) no-one suggested how the symptom might be used to diagnose the problem - all just offered general debug suggestions.

Q4: most students attempted this - good answers, in general. Most students who lost marks did so by entirely failing to address sub-parts of the question.

Q5: again, most students attempted this. For full marks to part (d), I would have expected a definition of what “fair access” means, as well as a plausible solution that stands a chance of working.
This year cohort was a fairly large one (60 students). The overall performance was slightly worse than in previous years. As a general comment and similarly to previous years, simple book-work questions proved to be surprisingly challenging, as opposed to questions asking for 'practical' examples.

Question 1. This question was taken by almost 80% of students. In part (a), a number of students described data warehouses instead of data marts, and thus lost some of their marks. The first bit of part (b) asked about making association rules from an already mined frequent itemset – marks were not allocated for the description of the process of getting frequent itemsets. In the second part, only the rules that could be extracted from the given itemset were considered. A small number of students considered rules like \( \{X,Y\} \rightarrow \{X,Y,Z\} \) which do not make sense. Part (c) proved to be a challenge, with many students failing to provide any specific features – there were numerous discussions as how the task of finding features is difficult, but rarely any specific ones have been suggested. There were also some misunderstandings of the role of training data.

Question 2 was taken by less than half of all students. Part (a) was a typical and simple bookwork question, but with the range of unrelated answers for which there were no marks allocated. MOLAP do not stand for multimedia OLAP and DOLAP for distributed OLAP. There were no problems with Part (b), whereas part (c) proved to be challenging for many (despite part of it being bookwork) with ad-hoc and unrelated definitions of the concepts. For some reason, information retrieval was sometimes confused with other concepts (e.g. OLAP). As a result, the students did either extremely well on this question or very poorly.

Question 3 was very popular (taken by 50 students) but there were a number of issues here. Part (a) was mainly OK, but with some students completely missing the point (sampling is not aimed at producing false positives!). Part (b) however was a real surprise: the task was to cluster the given dataset using hierarchical clustering and a number of attempts were combining k-means (!) and “ad-hoc” clustering, merging as many data points as possible in a single step. This was disappointing and many marks have been lost here. Part (c) was generally ok, but a number of students failed to provide any specific comparison between snow-flake and star schemas, mainly providing their definitions only.

Question 4 was taken by half of students. Part (a) was bookwork, but some students provided just a general discussion on data handling without specifically referring to data profiling, and marks were lost there. Surprisingly, part (b) turned to be difficult, with many students failing to explain how (that specific) meta-data (text summaries and ontologies) can be generated and used to support task. There were very vague references to meta-data (“ontology is something that is related to something”) and discussions of ontology-based annotation were very much superficial (“annotation is a difficult task”). Probably the major surprise and disappointment in this exam was part (c), where a number of students failed to describe anything that would look like k-means clustering and – again – this was a bookwork question. Some success was demonstrated in getting the quality of clustering right. Still, there were examples of high marks achieved in this question.

Question 5 was not a popular choice and many students got lower marks here. Part (a) was reasonable, although many answers failed to give a deep comparison between the two so some marks were lost there. Part (b) in general did not have any specific problems (although I have expected a bit more focused answers in part (i) rather than explaining the whole theory). In part (c), many answers were too generic and explained the association rule mining process rather than data-specific (ranges, the numbers of values, etc.) and Apriori challenges. Still, there were some very good answers.
Question 1

a) The question specifically asked about a model-driven approach in the real-world. Many answers did not take into account that in real-world there is usually more than 1 PSM and some hand-crafted code is required in the final implementation.
b) Mistakes made in the offered solutions included: user owned by question or answer (means that a user can only post one question or answer), no containment, missing string attribute to hold question and answers, inclusion of ids (not in a pim).
c) Generally well answered.
d) The question specifically asked for examples to illustrate answers, many answers included no examples. This question is about the style of generated APIs, it was not about, as many answers assumed, when the API is generated. The main points should be that an early bound style allows compile time checking of API calls; whereas, a late bound style requires runtime checking but supports the development of generic tooling.
e) The question specifically asked how a meta-model is different from other models; this was omitted from many of the answers. Many answers also claimed that a meta-model is more abstract than a model; this is not the case, it defines the concepts that can be used in a model.

Question 2

a) Too many of the student answers for this question gave generic answers that did not consider the specific case described in the question.
b) Again too many students failed to use details from the case described in their answers.
c) Yet again, answers did not take into account scenario described.

3. Almost everyone tried this question, and almost everyone got a good mark, although there were very few first class answers.

A) Most candidates picked up on the different “frequency of change” aspects of both scenarios, and recommended methodologies accordingly. However, another aspect in each case was the degree of novelty associated with the project - i.e., the degree to which the project was a standard engineering task or whether it required design of a radically new software system. Few candidates considered this, or if they did referred to it in only a vague way, which was perhaps the main cause of lost marks.

Other low-scoring answers were ones that listed the positives and benefits of the two approaches in general, but without linking them to the details of the specific scenario. An example of this would be a claim that agile is better because it doesn’t produce unnecessary documentation, without saying why this is of relevance in this particular scenario.

Finally, although not a major cause of lost marks, a very common misconception evident in answers was that, in scenario 1, the presence of the documented laws that the system must follow meant that a specification already existed. Although it is true that the documented laws provide the heart of the application semantics, there are many other aspects, to do with the training side of the system, that are not specified by the laws. Also, the laws will contain many aspects that cannot be automated in the tool, and which will need to be transformed into something that can be implemented in software. So, at best, we can say only that the laws provide a useful starting point for the specification. There is still a lot of specification work still to be done.

B) Almost everyone gave the obvious answer: final year student, recent graduate, employer. A few candidates gave some extra roles, such as recruiter, broker. Only a handful of candidates lost marks on this question (even though the generic “employer” role could be argued to be a bit too vague and unhelpful), for suggestions such as: agile team member and Web developer. The roles to be identified here are those relating to the use and long-term management of the system, not to the development or
C) The same kinds of problems with stories that were mentioned in lectures cropped up here in candidate answers. A small number of stories were under-specified, even by user story standards. For example, giving a function of "find the top employers". In some cases, the functions didn't describe something that the user wanted to do *with the software*. For example, a function of "As a graduate, I want to find a job so that ...", rather than "As a graduate, I want to see lists of employers in my area with open vacancies so that..." The first of these is something the graduate wants to do, the second is something the graduate wants to do with the help of the software.

The most common problem with stories, unsurprisingly, was with the business goal/benefit. Again, all the problems with goals that were raised in the lectures were present. To mention just two, it was common for goals to be underspecified and vague about the actual benefit, and for goals to actually be a function corresponding to the next step in some process. Here are examples of these two kinds of error, respectively:

Under-specified function:

"As a graduate, I want to see a list of employers with open vacancies in my specialism, so that I can find a good job"

instead of

"As a graduate, I want to see a list of employers with open vacancies in my specialism, so that I can save time by focussing only on vacancies relevant to my skills."

The second makes clear the benefit to be achieved by the software (the user will save some time), whereas the first just describes a woolly "good thing" that is not directly connected to the function.

Function given as goal:

"As a graduate, I want to see a list of employers with open vacancies in my specialism, so that I can submit applications for them."

Here, the thing that is given as the goal is actually a subsequent step in the process of finding a job (find relevant vacancies, select vacancies to apply to, submit applications, ...). It's another function that needs to be supported by the system, rather than a description of how the user benefits from implementation of the function in question.

Finally, the most common cause of lost marks for this question was the large number of stories that were given that were not epics. It is possible that some candidates misunderstood the notion of "epic" as meaning "a story that is not implementable in a single iteration". In fact, an epic is a story that describes a major chunk of functionality - something that will need to be broken into many stories, which will probably themselves need to be broken down even further before they can be scheduled during iteration planning. So, "searching for employers with open vacancies" is not an epic, even though it might need to be broken down further before being added to an iteration plan. "Matching final year students and graduates to placements based on skills and aptitude" is an epic, on the other hand, since it would involve stories to allow prospective applicants to register interest, stories to gather the information on skills and aptitude (perhaps stories requiring applicants to take on-line tests or organising phone interviews), stories to capture the requirements of the different placements, stories to perform the (possibly complex) matching process.
Quite a number of candidates tried to artificially create an epic story from a non-epic by using the word "manage" in the function. It was often unclear whether the resulting stories were genuinely epics. Stories such as "As a student, I want to manage my profile, so that ..." did not receive credit as epics if it became clear later that by "profile", the candidate meant simply a few standard attributes, such as predicted/actual degree class, degree topic, location. But, "as an employer, I want to manage my placements" is an epic, if managing placements includes the strategic planning involved in deciding what placements to fund, and in which teams, for example.

A smaller number of candidates tried to create epics by creating compound stories, where the function just listed a big set of things needed to be done. E.g. "As a graduate, I want to see lists of employers with open vacancies and to apply to them, so that ..." This is not an epic - it's just a poorly written (compound) story.

When awarding the marks for providing three epics, I did not give the benefit of the doubt. Stories had to be clearly epics or have clear potential to be epics, to earn marks, since this requirement was clearly stated in the question.

D) Refinement is a very difficult task, and so it is not surprising that many people struggled with it. A major cause of lost marks was an apparently random and incomplete split, where some aspects of the epic functionality had been converted into smaller stories but others had not - and where the choice of which stories to include appeared somewhat random.

A great many candidates attempted to split by operation, but in a rather mindless way. As we discussed in the revision session, a split by operation does not necessarily result in four more detailed stories, one for each operation type. Instead, several stories may be needed for each op type. And the stories that result may not actually contain the name of the operation type within them. So, a "delete" operation story may be worded as "As a graduate, I want to cancel my applications to one or more placements, so that ..." The operation needs to be put in the context of the overall semantics and vocabulary of the application being specified. Few people did this successfully in their answers.

Some candidates split up their search stories by creating individual stories for searching on specific search fields (search for employer by location, by industry, by salary, etc.). This could be justified, if a certain form of search had complications (such as implementing a "near to" function on a by-location search field). But in many cases, it seemed like an artificial split too far, since adding the search field in would just mean adding an extra column constraint in the database query that answers the search. The splits ended up looking very artificial and unconvincing (and no argument for completeness of the splits were given).

Many of the stories produced in the refinement suffered from the problems noted in the feedback to part c).

Finally, a handful of people misinterpreted the request to give one story which could be implemented in a single iteration, and instead gave estimates of how long it would take to implement each of the refined stories in elapsed weeks. This was surprising, given the emphasis in agile on avoiding absolute estimates of duration for stories, and earned no marks.

E) There were very few good answers to this part of the question. The major cause of lost marks was under-specified, non-repeatable test cases, where either some key steps were omitted (usually to do with setting up the fixture - the starting state for the test) or steps were not described in terms of actual values (e.g. "enter a first name" rather than "enter
4. This was a challenging question, though the majority of candidates attempted it.

A) There were many reasonable answers to this question, and a handful of really excellent ones. Marks were lost by candidates who addressed only the symptoms of the problems, rather than diagnosing and attempting to correct the root cause of the problems. For example, a number of people said that team D should fix defects as soon as they are discovered. This is just too superficial a solution. What was needed was to look for the reasons why the failed tests are being ignored, and to propose actions that could address these reasons.

Another misconception that appeared in several answers was a failure to distinguish TFD (test-first development) from TDD (test-driven development). Neither technique is mandated in agile, although an agile team which doesn't do some TFD would need to justify itself carefully. But it is certainly possible to use agile methods without applying full blown TDD. So, it is not necessarily a valid criticism of the teams described in the question that they are not using TDD.

One candidate suggested that team A should address its problems by faking test successes by hard-coding results into the code. Needless to say, this answer did not receive any marks. Faking test results is *not* a recommendation of any agile method that I am aware of, and is not part of TDD.

B) This was a challenging question, given that candidates hadn't had much chance to practice these techniques during the course (other than in the TDD lablet). I therefore set a generous marking scheme, to compensate. Unfortunately, there were very few really excellent answers. Marks were lost for the following reasons:

* The tests provided are reasonable in themselves, but don't actually describe behaviour related to the payment of the fine. For example, tests of the code which increments the library user's fine over time is not a test of payment behaviour.

* The tests were tests of GUI functionality, rather than of the interface to be provided by the Payment class. For example, some of these tests described logging on to the system, and entering values into fields. In these cases, I marked only the parts of the test that related to the behaviour of the Payment class, as viewed through the GUI.

* The test cases did not set up the fixture for the test (i.e. the starting state). For example, a number of solutions wrote test code for making a payment without first setting up a library user with a current fine. A number of other test cases used uninitialised fields on instances to execute the test.

I did not mark down candidates who chose not to have a separate Payment class, and instead designed the payment functions as part of the interface of some other class (such as Library User). As long as the test was directed at core payment behaviour, marks were awarded in full.

There were also a number of odd modelling choices, resulting from the test cases. For example, some people chose to put the details of the credit card for payment on the LibraryUser class, meaning that the library user must use this one card for every payment (until changed), rather than allowing a different card to be specified for each payment. I was also surprised by the prevalence of get/set methods that required the field name affected to be given as a string value as the first parameter.
A very small number of candidates (thankfully) seemed not to know what a test was, and instead wrote small chunks of production code.

C) There were a couple of really excellent answers to this question, but I'm afraid that many answers were rather lacklustre. Marks were lost for: making the same point several times, but using different words; making the same point for agile testers as had already been made for agile in general; waffly answers that talked about specific characteristics of agile as needing courage, but without explaining why; mentioning features that needed courage but which are also a factor in any development, agile or not. An example of this last kind of error was to talk about the courage needed for testers to write tests that really exercised the functionality being implemented. This does indeed require some courage, but it is just as true for testers on more convention projects as for agile testers.
In general I am happy with the way the exam panned out. There were no questions that were
universally avoided, which indicates that there were no complete blackspots in what I taught,
and none of the questions was perceived as being impossible. Every single student chose to do
Q3, which would suggest that it may have looked particularly easy, but since that's not the one I
thought would be especially popular then I don't mind.
No single question got particularly high or low marks, and the spread of marks overall looked
OK to me—a couple of firsts, a couple of very weak ones, and lots in between. I haven't done
means and standard deviations, I don't know the cross correlation with other courses taken by these
students, but at first sight it all looks OK.
There were a couple of questions where I altered the mark scheme in the light of the way the
questions got answered—one where lots of students read the question in a way that I had not
intended, one where there was an easy answer that worked just as well as the more complex
one I was looking for. In both cases I gave marks for what the students did, rather than what I had
wanted them to do. I still think that the obvious reading for question (1a) was the one I intended, but I
can say that it could have been interpreted differently. The fact that Q3 had an easier solution
than the one I intended is just annoying, but since the answer they gave worked OK then I have to go
with it.
(1a) The answer I wanted was about all the things you have to do before you arrive at a
parse tree—look the word up in the dictionary, apply spelling rules at morpheme boundaries,
combine roots and affixes, POS tag, parse. A large number of candidates read it as being
purely about parsing. Looking at the wording, I can see how they arrived at this reading, so
I've marked it either way. However, in order to get a good mark when answering it as being
about parsing I have required a correct and detailed trace of the parsing process AND some
discussion of alternative algorithms (or at least a recognition that there are alternatives).
Otherwise it's a very easy 10 marks—just sketch the parse of a fairly simple sentence.
(1b) Including the sentences <det>the<adj>smelly<noun>child<verb>smiled and
<det>the<adj>smelly<noun>child<verb> turned out to be slightly misleading, because some
candidates decide that those were the examples I wanted the regexes to cover, whereas I
actually included them just to show how I was expecting tags to be inserted. I've made
allowance for that, though almost everyone who did that completely cocked the question up
anyway.
3: I wanted them to use syntactic-relatedness as the context in the practical part. Quite a few
of them used sentences or paragraphs. Unfortunately, that actually worked just as well, so I
had to give them full marks. It meant that the question was a bit easier, and a bit more
boring, than I had intended, but I can't knock marks off for that. However, I have been firm
about not allowing full marks unless they actually showed me the vectors, or did something
else fairly detailed.
Ian Watson:
Q1. Overall reasonably well answered, but the following should be noted.

a) Several answers ignored the request for the use of the code for a binary semaphore and used other code - this lost marks.

b) Most answers simply said that serial code could not share variables. The answer required was that the interleaving of instructions can occur on a process switch in exactly the same way as in multi-core. However the solution is simply to turn off interrupts in a critical section to make sure the timer does not cause this to happen.

c) Some failed to emphasise the atomicity of the instruction.

d) Many did not state explicitly that this is a read-modify-write instruction and so must make two memory accesses. Only a few discussed the issues of implementing this in a RISC pipeline.

e) Generally well answered although some just gave the code with no proper explanation.

f) Again explanations of operation were sometimes poor.

Q2. Generally answers were good

a) Few discussed background, i.e. the need for each processor to have a cache and the issue of the expectations of a programming model.

b) Most got this right.

c) Explanations were generally good although some rather brief.

d) Many got this correct. The most common mistake was to think that the add instructions changed the state of the cache.

e) Most observed correctly that the answer was x+1. The correctness issue was, however, poorly discussed. The answer was that the intention of the program was probably to end up with x+2 which is what the answer would be if the two sections of code were not interleaved. But the code as presented has a data race and therefore locking would be required to avoid inconsistency.

Chris Kirkham:
Q3: Everyone did this - but the answers to part ©, the problem, were generally poor. Too many answers showed a failure to understand "fine-grained locking", which here means having locks for individual elements of the array. Of those who did understand this, too many tried locking ints - the right answer is to have a corresponding array of locks!

Q4: There were only two answers to this question, probably a sensible decision as it is not easy to write a good answer to part (b). Both answers really missed the point of part (a), however, which was to illustrate why locks on Objects are not sufficient by providing examples which actually needed Locks from java.util.concurrent.locks.
David Rydeheard:

Section A (Q1 and Q2). On the whole the answers to these questions were not strong and the overall average for each question was c51%. Approx 90% of the students did Q1, the remainder doing Q2.

Q1: String matching algorithms:

Part (a) KMP algorithm (10 marks) - this is standard material, and many marks were lost through imprecision in describing both phases of the algorithm. It is not simply a matter of finding a maximum common prefix and suffix, but how to restart after a shift. The algorithm is linear O(N) - approx half the students got this, but few explained fully why this is.

Part (b). This is NOT the BM algorithm. The 'bad character' shift of BM works only with the other shift calculation. Without the second shift calculation, 'bad character' shift may cause backward movement of the pattern! Some marks were awarded for explaining the 'bad character' shift, but full marks only for a correct algorithm. Almost no-one answered the last part of the question - and many didn't read it properly, so gave irrelevant answers for no marks.

Q2: Graph algorithms:

Few students answered this question, which was not particularly difficult. Marks ranged from very good 14-16 to poor 5-8. The first part was to explain lectured material - most got good marks on this (but a few clearly didn't know the required algorithm).

The topological sort: Many answers had the core of a correct algorithm, but marks were lost through imprecision.

The exponential algorithm for an NP-complete problem: Again most attempted this, but lost marks either through not explaining clearly what the problem being addressed is, or through unclear explanation of an algorithm.

Section C. Overall, the answers were not answered that well.

David Lester:

Section B.

Q3 Attempted by 60/65 students. In the majority of cases this question was answered very well.

Part (a) being a definition caused some problems, and I have awarded 2 marks for any sensible informal definition. Unfortunately, most non-formal attempts to part (a) were circular, i.e. "f is O(g) iff f is asymptotically g", and scored no marks.

Part 3(b)(i) Again mostly well done. A number of students who omitted the formal definition of part (a) failed to note that we require a "when n is big enough" component to the argument and they received 3 marks if the rest of the answer was correct.

Part 3(b)(ii) caused some problems. In the end I accepted any of the following answers: O(g_1+g_2), O(g_1), or O(max(g_1,g_2)). Again, a number of students omitted the "when n is big enough" component, and they lost a mark.

Part 3 (c) Was usually well done. Most students accurately placed O(2^n) \subset O(n!) \subset O(22^n). A number omitted Stirling's formula; in a handful of cases this didn't matter as an induction proof sufficed.

Q4 Attempted by 3/65 students. Two got full marks and one got 1.
Howard Barringer:
Q5, which was attempted by 19 of 63, achieved an average mark 50.5% but had a spread from 1/20 to 19/20, with the majority of marks in the range 9/20 to 16/20. Answers were in general terms sloppy and not precise. As an example, part (b) asks for a definition specifically in formal terms of a factor f approximation algorithm (this is standard bookwork given in the lectures) but many answers gave an informal and inaccurate description. Sloppiness arose in answers to part © as well, with few failing to note that an edge-weighted tree is being handled and that the required and steiner vertex sets are disjoint. Most answers were acceptable for parts (d) and (e). However, very few answers attempted part (f) and of those that did none gave the geometric construction for the Steiner point when the nodes form a acute-angled triangle.

Q6, attempted by 44/63, had stronger answers overall, with an average mark of 55%, had a spread of 3/20 to 16/20, with 40/44 answers in the range 9/20 to 14/20. Parts (a) to © were answered mostly quite well, although a common error in part © was failing to explain what the expectation of the random variable X, denoting the number of iterations to the first correct solution, was. Part (d) received few answers and those that did mostly failed to indicate (1) how a contraction was done using a variation of the simpler randomized min-cut algorithm and (2) that, in the step case of the recursion, the graph should be randomly contracted to the ceiling of $1 + n/\sqrt{2}$, twice, and why!

This paper was -- in the main -- very well done. As usual marks were lost for inaccuracy or inattention. Typical problems were: failure to give all cases in part (a); omitting one or other of the two proofs needed in part (b); and failure to address the question properly in part (c).
Comments:

Q1 was answered by 33% of the class, average mark was 45%
Q2 was answered by 67% of the class, average mark was 65%

General comments

1. Many people ignored the instructions to use diagrams, and what diagrams which were given were often extremely small and/or shabby. This is a perennial problem. Where people did give illustrative sketches, these were mostly straight from the lecture notes. This is acceptable, but original examples would have been preferable (surely people could think of something other than the fish).

2. Some answers were exceedingly brief, with little coherent narrative explanation. Such answers cannot be marked as highly as answers which are "readable".

Q1.
* Overall, not very wellanswered (avg: 45%).
* The main loser of marks was incorrectly drawn/explained geometry of the laser scanning system. This is surprising, because the geometry is very simple.
* It was clear that some people had not read any of the "required reading" papers (and one candidate wrote that he wished he had).
* About 2/3 of people did not appear to know what "semantic structure" meant when applied to polygonal meshes. I specifically spoke about this topic in at least one of the lectures.

Q2.
* Overall, quite well-answered (avg: 65%).
* The least well-answered part was the explanation of extracting geometry from video sequences.
* Some very poor, scrappy diagrams. A small monkey could have done better… what is the problem with drawing a large clear diagram?
* The part about puddles was designed to stretch you a bit, and think about things you had seen on the course (I don't believe we covered "puddles" specifically!). Most people did quite well on this part, which was pleasing.

Question 3

Generally the question was well answered, with the majority of those who made a serious attempt at it showing a good understanding of the topic.

- The question explicitly asks for an explanation of primary rays, secondary rays and shadow feelers, and for a diagram. Numerous marks were lost by omitting one or more of these components, and instead giving a general overview of raytracing.
- Of the diagrams that were drawn, many were tiny scrappy doodles, poorly labeled. Make diagrams bold and clear -- there is no shortage of paper!
- In the final part of the question where a comparison between radiosity and raytracing was required, many students missed the 'obvious' difference that one is viewpoint independent -- this point was made numerous times in the lectures.

Question 4

As with question 3, many marks were thrown away by producing diagrams that were lacking in detail, or were (commonly) just too small to see (some the size of postage stamps!)

Typically though the question was well answered, with no specific pattern for lost marks beyond 'lack of detail'.
Overall the students’ performance was very satisfactory. The average mark for the paper was 61% and the average total mark (after adding the coursework mark weighted 30%), was 62%. Out of 48 students, only 5 students got less than 40%. Of these, 4 got a very low coursework mark (0%, 0%, 4.17% and 12.5% respectively), evidence that their poor performance was due to lack of work rather than inability to cope with the material.

The questions proved appropriate, judging by the distribution of results. Four issues require some adjustment:

a) Question 1 is compulsory and in the form of a test of 10 items. The students were given many examples and urged to answer very concisely. Yet they very often ignored this advice, and so spent more time on this question than was appropriate given the marking scheme. They will need more guidance on this particular point. The ability to summarise the essentials of any point is an important skill.

b) In two questions some given lines of code were highlighted (using boldface type). On some papers this was poorly rendered. I propose to mark such lines by an asterisk in a separate column in future, or some such device.

c) Some parts were incorrectly numbered, I think as a result of revision of the paper. This did not affect answering but showed poor standards. I accept full responsibility for that.

d) In two questions the students were asked to reconstruct a piece of code after this had been jumbled up. The wording I used to ensure they would rewrite the full original text (rather than use reference numbers) was not understood by a few students. A better wording will have to be worked out. Note that this kind of question is very effective, if constructed properly.

I would like to add that this year, the number of lectures was reduced to 11 hours (from 22 and more in the past). I feel this did affect the delivery of the course, which I feel is unsatisfactory.
In total, 62 students sit in the exam.

Question 1: taken by 58 students, and the average mark is 57%;

Question 2: taken by 44 students, and the average mark is 66%;

Question 3: taken by 21 students, and the average mark is 54%;

Question 4: taken by 38 students, and the average mark is 64%;

Question 5: taken by 25 students, and the average mark is 55%.

The overall average mark = 59%.

5 (out of 62 students) fail to pass the 40% mark; 28 (i.e. 45%) students get results >= 60%.

Question 1:

The most common mistakes are in 1(a) and 1(c).

In 1(a), the most common mistake made by most students is that they failed to make clear what measures should be used to design a protocol that is the most efficient.

In 1(c), most of the students attempted this question focused on how to distribute keys, rather than designing a signature protocol using a symmetric cipher and an arbiter.

Question 2:

Students did this question mostly well, but the common mistake by some of the students was they muddle up off-line password guessing attacks with on-line guessing attack.

Question 3:

This one seemed to be the hardest question for the students. The most common mistakes were in (b) and (c), where many students could not explain how counter-mode encryption works and how Cipher-Block-Chaining Message-Authentication-Code works. These topics were covered during the lectures.

Question 4:

The bit that the students lost most marks was in (c). This question requires thorough understanding of several cryptographic primitives and important concepts, including how DH works, what is the security weakness of DH, what are the countermeasures, the need for PKI and digital certification, and digital signatures.

Question 5:

Most students lost mark in (c), and the common mistake was that they failed to make clear: an important idea for the protocol design was the need for a trusted third party T to assist in this fair trade, and by introducing the use of T the design of the protocol should not introduce...
further security loopholes.