Q 1. This compulsory question yielded a slightly lower than expected mean mark of 51%. 14% achieved firsts, 56% 2ii or above, but 22% also failed. The vast majority were able to simplify expressions using big-oh (part a), and also did well on recognizing the relative growth rates of functions. However, they lost marks on interpreting a log-log plot, and using it to infer complexity. Many also performed poorly when asked to state why an algorithm was correct, and derive its complexity from pseudocode.

Q 2. Hashing: Over 80% of students did this question, and most performed reasonably well with an average of just over 60%. Most provided exact answers for the hashing examples using various collision resolution techniques. Most also appreciated that the worst case for linear probing is linear - but some lost marks in not appreciating that it is linear in the number of elements (not hash table size). The deletion algorithm was not done well - very few got anywhere near full marks. Deletion of an element cannot be achieved by simply removing the element - other elements have to move (the hint was not used by many!).

Q 3. Trees: Relatively few people attempted this question. There was a split between the marks. Students either did rather well on this question (14-17) marks, or performed poorly (3-5 marks). Most of the students did the first part of the question (insertion into an unbalanced binary search tree), but struggled with balancing and the AVL trees. Relatively few people attempted the advanced part of the question on Euler's traversal and the common ancestor of two nodes.

Semester Two

Question 1 Dijkstra's Algorithm

i) I was looking for 5 aspects of the algorithm: initialization of the priority queue, the loop (especially the stop condition), pulling the best out of the priority queue and checking its children, comparing the distance to the stored distance, and calling the change function.

Many people showed good understanding of the algorithm, and got all of these points (or maybe forgot initialization). Those who did not get the points showed no understanding of the algorithm.

ii) Many people got full marks for this, or lost one point for incorrect details. A few people wrote down an all to all distance matrix, which I could not understand. I gave no marks for this. I assume this was a confusion with dynamic programming

iii) I was looking for four things: the two correct complexities, setting them equal and one for solving the equation for the result. Many people did not know the complexities of the two algorithms.

iv) People could not state clearly what a heuristic is, and many people could not say when it would be appropriate to use one. Most people who attempted this got some points.

v) Very few people thought of the obvious heuristic, which is: if there is a direction connection to the target, take it, otherwise go the city which is nearest to the target. Therefore I gave credit for any valid heuristics.
Question 2 Knapsack

Large majority chose the question. Overall, the distribution of marks is right.

Students on the whole recalled the knapsack problem itself very well. Most clearly had taken advantage of the learning opportunities (tutorial, lab, lectures, practice exam and textbook). They also explained the greedy algorithm adequately and almost all knew it did not give a guaranteed optimum. One common error on the greedy method was stating that items should be ordered by value, or worse by weight, instead of by the “benefit” - the value to weight ratio.

On the dynamic programming parts, students worked through the given algorithm well - about three quarters of the answers filled in the DP table correctly. Perhaps a quarter of them were able to work out the reduction in the last parts. This required spotting that one problem was a scaled version of the other, and so could be reduced to the other.

Question 3 Sorting

Large majority chose the question. Overall, distribution of marks is right.

There was good recall of the time complexities of the main sorting algorithms. Almost everyone got at least 4 out of the six algorithms correct and probably half got all of them.

Almost all students chose to define “stable sorting” and did so correctly. Almost all chose to define “comparison-based sorting” and many did manage to write something additional like that it is general-purpose or that there is a lower-bound on complexity. Many students defined “in-place” correctly, but about half said no additional space (rather than a constant amount) is used. The definition of a “total order” was less well done.

The calculation of the time complexity of the given radix sort algorithm was mostly ok but very few managed to remove M, word length, as something that does not normally grow! Space complexity was slightly disappointing as few remembered to count only the additional space used and few recognised the bucket array needs O(n) space. Attempts at defining correctness were weak overall, but contained elements of an argument. Most got 1 or 2 out of 4 on this.

Question 4: Trees and hashing

This question was answered by 35-40% of the total number of students that took the exam. Most of the students answered the question correctly, especially the technical part of it showing how to insert/remove a new node into/from a heap.

Some small problems arise in the part of the question related to the asymptotic complexity of these algorithms, with some students missing the point that it should be O(log n), stating instead O(n) or O(n log n). Overall, I feel that they did well on this question.

Question 5. Graphs

An average number of students took this question. Whilst some had very good marks, the general performance was poorer than average.

Those who knew the basics about graphs (representations, applications, traversals, for example), tended to lose marks by not fully answering questions, for example by not discussing complexity issues where necessary. The description and algorithm for DFS was in general answered well. The final part (connected components for undirected graphs) was answered poorly in general: the notion of a component was not properly understood by many.
Comments: A special resit examination - no feedback provided.
Q1
a-e): No particular problems. Many students got full marks. In d) several students tried to explain the terms, but the question was only asking for an example.
f) No comment.
g) The question was not about what "at-most-once" semantics meant to the client, but about what the server had to do to provide that! h) "Idempotent" does NOT mean that an operation is done many times!
i) No comment.
j) The question was not about what a digital signature was, but about secure digests. Too many waffled about the purpose of including one in a digital signature (e.g. "to improve the efficiency and effectiveness of the security provided") - without gaining any marks!

Q2.
(a) (i) The question used the term "in detail", so it was necessary to mention the use of queues for requests that could not be satisfied immediately, and that a client granted the mutual-exclusion token has to return it.
(ii) The simple algorithm described in the question is NOT the BullyAlgorithm! It requires 3(n-1) messages, if there are n processors in total. The point is that the simple ring election algorithm as a worst case needs 3n-1 (or perhaps 3n-2), which is worse. However the best case and the average case are both better! Some also took advantage of the context to simplify the basic ring election algorithm, using the fact that only one process would initiate an election, and therefore it could notice when the election message reached it - and announce the winner. They got the marks, as did those who failed to notice that possibility.

(b) (i) "File update semantics" is NOT the same as message delivery semantics! So there were no marks for discussing "at least-once" versus "at most-once", etc. And students lost marks by explaining "client-side caching" without relating it in any way to "file update semantics!"
(ii) Too many students explained what a timestamp was in general, and failed to relate it to distributed file systems. A good answer should explore the trade-offs between using "call-backs" and using timestamps in this context.

Q3
38 out of 90 students answered this question. A) many students failed to see that it is the rate of arrivals in conjunction with the completion time that make it impossible to provide guarantees. B) Generally ok, although several students failed to describe something that could be used as an algorithm for load balancing.
C) Different answers were given, not only using Amdahl's law that was mentioned in the lectures. Marks were given as long as it was clear that the sequential part (5/20) was the key factor determining the bound.
D) Many students failed to see that this question wasn't about proving that "always" it is possible to act in agreement when the number of generals is less than 3 times the number of traitors! This is a well-known solid result and it is only that if the traitors don't act intelligently, the loyal generals may still make an agreement in "some" but not all cases.
E) Many students considered only the possibilities for execution of the different statements of the two processes. However, consistency relates to the values on the replicas!

Q4
62 out of 90 students answered this question.
A) Generally ok. Some students didn't show an execution sequence or didn't consider Tom's messages only.
B) Generally ok.
C) Many students failed to realize that a crash can occur at any point in time!
D) Again this is used by the server to coordinate between the database and the MOM. The server doesn't have to coordinate anything with clients; the clients act on their own!
E) A tricky question, not well answered. This semantics is useful for handling messages at
the client side, but the questions were about the server side!
Section B

Question 1
Mean was 50 ± 24 %
Not well answered. Very few answers included everything that the question required. Although there were a few very good answers, there was an equal number of poor answers. Disappointing as this was mostly a bookwork question, with a small amount of application.

Question 2
Mean was 51 ± 17 %
Much better answers, the students mostly understood the invariants in the image and appreciated simple methods of exploiting them. Generally very sensible attempts

Section C

Before I look at each question, I want to pull out some observations that apply equally to both:
• Many people completely ignored the clear instructions to "illustrate your answers with appropriate diagrams", and instead just wrote text. Not surprisingly, they lost marks. You would think that it would be obvious that answers to graphics questions need graphics. Apparently not.
• Some students wrote jokes in their answers, or drew silly pictures. This is inadvisable, especially for those whose understanding of the topic is poor.
• Some people did draw diagrams, but they were often either so tiny, or so poorly drawn, as to be useless.
• I was astonished at the amount of handwriting that was simply illegible. Anything that the examiner cannot read, clearly cannot score any marks. Believe me, I do make an effort to decipher handwriting. But there comes a point when you decide it's illegible, and move on.
• One person deliberately answered two questions from Section C, and wrote me a note asking me to mark both, and award him the higher of the two marks. This is discourteous, selfish and unpleasant. I refused the student's request, as University regulations stipulate: "where a student has completed too many questions, the required number of questions will be marked in the order they appear on the paper, with subsequent answered questions being disregarded.
• I got the strong impression that students answering these questions belonged to one of three groups: those that came to most of my lectures; those that came to some; and those who didn’t attend any. Not a particularly enlightening analysis, you may say, but I bet I could sort the papers into at least the latter pile. My weekly attendance headcount was never above 46%.

Question C1
Average mark 7.0/20 => 35%
stddev 4.6

This question had the worst average I have ever seen on one of my papers. The question was basically in two parts: the first (Sections (a), (b) and (c)) covered material directly addressed in the lectures and the course notes. This was answered reasonably well by most people; the second Section ((d) i-vi) also addressed material I had covered, but asked you to think, and to synthesise your knowledge to address a particular scenario (the digitisation of the British Museum’s collection). This section was in general answered very poorly. Only one person referred to AC3D, for example. Nobody at all related the question to the coursework you had done. It was as if that part of the course was in a completely different part of your mind, inaccessible during the exam.

Question C2
Average mark 8.7/20 => 43%
Students answers to this question fell into two categories: those who understood the material, and were able to describe what was going on in a sensible way, telling a story; and those who had simply memorised the material, and were unable, in their answers, to demonstrate any understanding whatsoever. Regurgitated material scores marks, of course, but it cannot score full marks. Utterly clueless regurgitation scores even fewer marks. Some students presented their knowledge of computer games as some sort of indicator of their understanding of the material. Alas, this was in all but one case completely misplaced. Game design is a discipline which has entirely its own set of constraints and ideals, almost none of which we addressed in this introductory course.

**Comments on COMP20092 – May 2010**

1)  
a) Most identified different access patterns suggesting different cache configurations, e.g. different associativity. Some mentioned bandwidth, but few remarked on the different positions in the pipeline and hence an obvious reason for physical separation.
b) Generally well answered, but quite a few forgot to discuss the relationship between cache size and speed and/or size and hit rate.
c) Most got the definitions right. Quite a few forgot to mention why maintaining consistency between cache and memory could be important. Many omitted to mention write buffering to alleviate the disadvantage of WT.
d) Most got the basics right. Some forgot to mention spatial locality as the reason why bigger cache lines may be better. Only a few mentioned memory burst mode for more efficient memory access for bigger lines. The most common mistake was to omit to discuss that there is an optimum due to limits of locality and displacement of useful data as the line gets larger.
e) Fairly poorly answered. Many failed to work out correctly how the address is divided into tag, index and data select. Hence, by getting the index wrong failed to gets the hits and misses right in part (ii).

2)  
a) Generally well answered.
b) Very few produced a good pipeline diagram. A bit more was expected than just ‘the fetch stage fetches’, ‘the decode stage decodes’ etc.. A proper pipeline diagram should have been drawn with the caches, register bank, ALU etc..
c) Generally well answered although quite a few wrote generally about control hazards without addressing the issue of exactly what happened when the branch is not detected until the execute stage. Note that this part of the question did not ask about branch prediction, so there was no point in discussing it here.
d) Many got the general idea but were lacking in detail. Only a minority drew a correct diagram of how a BTB works.
e) Generally poorly answered. Many got the general idea that the third instruction depended on the other two and there were various suggestions about inserting delays or NOPs etc. Very few drew a good diagram of the pipeline with the correct forwarding paths.

Q3 and Q4
By and large, the students have made a decent attempt at Q3 and Q4, even if they haven't done as well as I hoped. One or two got confused between multithreaded programs as opposed to multithreaded CPUs - I suspect they missed my lectures on the subject. I'll reword questions next time to make it blindingly obvious. I'm disappointed by the answers to the numeric problem parts - I'll set more of these as revision-tests in the future during the course.
Q1: This is a compulsory sampler question. The great majority of students achieved decent marks (>50%), with a good number close to full marks. A depressing number of students can't do basic arithmetic (e.g. work out the capacity of a CD in part g)).

Q2: This question was based on their laboratory project work. While many gave answers that indicated that they remembered the techniques that and their partners had used, very little of the experience gained was evident in the answers and very few students quoted any quantitative results.

Q3: This question as in retrospect too easy, although some students contrived to make more hard that it actually was.

Q4: This question was attempted by just under 50% of those taking the exam, with the majority performing badly (<50% mark) on it. Surprisingly few students were aware that motion compensation (which involves unspecified search) is the major asymmetric component of MPEG video compression, which was a problem for parts a) and b) (total 6 marks). Most could make up reasonable answers to part d), but few moted the similarity of options between parts e) and d).

Part A covered all topics in the form of Multiple Choice Questions. Because it was mandatory, all students had to solve it. The average mark was 70%, showing that students feel very comfortable with MCQ, where they have to choose the correct answer from a number of alternative answers.

Part B-Question 2 was an essay like question and was not mandatory. It covered the topics of extensions to SQL (PL/SQL), SQL and design alternatives for Relational and Object-Oriented Databases. Only 10% of the students sitting this exam attempted this question, and the average mark was of 55%, which is reasonably low. Most students failed to give a correct/complete answer to the PL/SQL part, which was Bookwork related, and to the Design Alternatives part, which required independent thinking and problem solving skills.
Semester One

Comments on question B1 are from Robert. All other comments are from John

Generally the marks, especially for the multiple choice questions, were very good. With hindsight the exam was rather too easy, but it certainly showed that most people understood the basics of Software Engineering. There were only two of the MCQs that most people got wrong.

Question 1.4 was

“This software has bugs in, we just don’t know what they are yet”. For which of the following components of the ABC system is this statement almost certainly true?

- The calculator we provide if an exam involves sums
- The code which stores your answers to disk
- The sophisticated tool we used to set exams
- All of the above

Most people automatically selected “All of the above”. Let’s think about the calculator. It’s basically mapping button presses to simple operations – one short method per button. So while it might have bugs in this is not “almost certainly” the case. Hence neither the calculator nor “all of the above” is the correct answer. Given that it shouldn’t be hard to figure out which is the correct answer and why.

Question 1.7:

Which of the following are advantages of the Waterfall development process?

- It allows effecting planning and budgeting at the start of a project
- It promotes specialisation of roles e.g. systems analysts, designers, programmers
- It ensures that problems are found early, when they cost much less to fix than later on
- None of the above

As explained in lecture 2 (see slide 7 in particular) all the supposed advantages of Waterfall proved to be illusory, and hence the correct answer is “None of the above”. The most plausible of the alternatives the specialisation one, is incorrect because of the problems which that specialisation causes – lack of communication and difficulty in deploying staff efficiently e.g. (pure systems analysts are only useful in the analysis phase). Modern software developers therefore tend to be involved in most or all of the disciplines of Software Engineering. (A consequence of this is that if you’re not happy writing code you’ll find it hard to get a job in software development). In section B a large majority of people chose Robert’s question rather than mine. Perhaps this was because Robert’s question was easier (it didn’t involve actually writing a use case), but I suspect the main reason is that generally people were happier with Robert’s material, and the way it was taught. This is something I will look at for next year.

Question B1

A customer can use the Acme on-line photo shop to create many types of product using electronic images such as those taken with digital cameras or from scanned “old fashioned” photos. These products include wall calendars; desk calendars; photo books and such like. The customer can choose a type of photo based product and then is taken through a series of steps to create that product, uploading photos as necessary as the product is designed.

The customer can modify the look and feel of the product with borders, patterns and other typical photo-editing techniques such as scaling and colour adjustment. As photos are uploaded, they are stored in a database for later re-use. The product itself can be stored in the system’s own database; it can also be downloaded in electronic form to the customer’s own computer, where it can be viewed, but not altered and uploaded again. Previously
created photo-products can be retrieved and possibly modified to be re-manufactured.

Once created and completed, the customer fills out an order form requesting number of copies; a postal address; etc. A bill is calculated and the customer pays on-line and the system uses a credit card payment validation system. Once the order is finished the customer's photo product is sent directly to a photo processing system where the product is manufactured. A human operative then picks up the printed calendars etc; and has it posted to the customer by a traditional postal service. The system then sends out an email saying the order has been sent.

Identify the actors in the scenario above 3 marks

Section B

1 a) (I) Many people found the correct actors. There were two main faults: Only recognising human actors and ignoring system actors. The second was to be deceived by the human operative who took things off the production line; as an indirect user of the system, he/she doesn't count. The sentence "As photos are uploaded, they are stored in a database for later re-use the product itself can be stored in the system's own database;" has two potential actors -- the photo database and the product database. The word "system's" should lead to one database not being an actor due to the "internal" nature of the entity. Assumptions have to be made about the other (ideally, of course, more questions need to be asked.)

Identify the use cases in the scenario. 4 marks

(II) Many use cases were too generic: "use system"; this one is too large grained as well, conflating too many issues. The opposite problem was also present -- too much detail. Using the actors and seeing what actions they perform is a useful check; similarly, cross-checking actions performed in the scenario with actors is another useful check.

For each use case, describe the participating actors. 3 marks

(III) this was generally answered well enough. the main fault came from technique: this question uses answers from (I) and (II) -- so making the answers consistent between these three parts would be a good idea. Cross-checking one's own answers is good technique.

State what assumptions you have made in your answer 2 marks

(IV) Assumptions were generally poor. A good assumption was to decide on whether or not the DB was an actor. There is insufficient information to do so without further questioning. Assumptions that the scenario is both complete and correct are not good ones. If you take the definition of actor etc., and then inspect your own thinking, it should be possible to work out where you've made some assumptions.

State the style of user interface that you would use for this system, and briefly describe this style. 4 marks

b) (I) Mostly answered well, but some described how they would implement, rather than the design style they would use. A brief relation of the user interface to the scenario would also have helped in some answers.

Briefly justify your user interface choice 4 marks

(II) The justification for the design styles was much less good. There is evidence and implication within the scenario that should point towards a certain style of design. Enumerating things like needing to access lots of functionality in arbitrary order "all at once" when manipulating an image would imply a direct manipulation approach. Having a constrained set of steps with an ordering would imply an overall wizard style; and so forth.

Question B2:

a) Explain why the set of classes on a design class diagram is not usually the same as the domain class diagram from which it was derived. (3 marks)
Very few people got full marks for this, largely because a lot of people failed to answer the
effect question asked, which was about the set of classes on the respective diagrams, not the
details of the individual classes. So you needed to talk about classes that appear on one
diagram but not the other, e.g. external software systems (domain not design) and pure
fabrications (design not domain).

b) For each of the following, state whether it is appropriate to show it on a domain class
diagram and briefly explain why.

i) Types of attributes  2 marks

Again a number of people misread the question – it says types of attributes – which are not
shown as they are software-related rather then just attributes, which are.

ii) Multiplicities on associations  2 marks

Most people correctly said that these are shown, because they are very helpful in
understanding the domain.

iii) Operations  2 marks

Again most people correctly said these are not normally shown – this is mainly to keep the
diagram simple and easy for stakeholders to understand. (Not all experts agree on this; and
there’s no great problem showing a few key operations without doing into details of parameter
types etc. The minimalist style of domain model I taught is based on guidelines by Larman
and my own experience.)

c). The Irwell Media Store started off as a bookstore, but later started selling CDs and DVDs
too. More recently it has started selling laptops too, and it plans to branch out into other
electronic devices such as phones and PDAs, although it does not intend to become a
general store. The company is taking its first tentative steps into the online marketplace, and
has commissioned you to produce a prototype inventory system

Draw a domain class diagram which represents the kinds of products which the store
currently sells, in a way which can be easily extended in the future. Hint: the best solution has
7 classes of which 3 are abstract.  5 marks

The ideal solution is a pure inheritance hierarchy with an abstract Product class at the top,
and abstract subclasses Media and Device (or similar). The Media class has concrete
subclasses Book, CD and DVD, while the Device class currently only has Laptop as
subclass, but the description indicates that others such as PDA may be added in the future.
Some people had the same general setup, but separated books from electronic media, which
was fine.

Quite a lot of the answers got full marks. A number of people had the right idea but used the
notation correctly (having the inheritance arrows the wrong way round, or using the wrong
sort of arrow altogether). I deliberately showed you only a small subset of UML class diagram
notation so that you don’t have much to remember in order to use it correctly.

d). If you were modelling a general store with thousands of different kinds of products, how
would the domain model have to be different?  2 marks.

It would not be feasible to have a class for each different kind of product. There would either
be a single Product class with some sort of product description, or Product could have a
small number of subclasses representing different broad categories of product.

Few people got the idea that the domain model would have to be completely different, not
just a much bigger version of the one in the previous part.

e). An inventory consists of a list of line items, each of which represents a particular product,
the number in stock, and the price per item. Suppose this is implemented as follows:

```java
public class LineItem {
```
private Product theProduct;
private Money UnitPrice;
private int numberInStock;

... etc.

and the Money class has methods to add two amounts of money or to multiply an amount by an int:

    public Money add(Money other)
    public Money multiply(int timesBy)

i). Write a method for the LineItem class to calculate the total value of a line item. 1 mark

    public Money getTotalValue() {
        return UnitPrice.multiply(numberInStock);
    }

ii). Show how you would write a method in the Inventory class to get the total value of the products in the inventory assuming that the list of line items is stored as an instance variable called items, and the Monday class has a public constant NO_MONEY which is £0.0. 3 marks

    public Money getTotalValue() {
        Money result = NO_MONEY;
        for (LineItem item: items) result += item.getTotalValue();
        return result;
    }

It was perhaps a bit mean to include some actual programming in the exam, but the point of software engineering is to produce software, and it was only four marks. I accepted less elegant solutions which worked, and ignored minor errors of syntax. The result was binary – people either could do it or they couldn’t, which was no surprise.

Semester Two

1. Part A of the exam comprised ten short questions.
2. A number of students did not attempt all the questions.
3. Some students, when asked to explain an acronym, merely elaborated the acronym without any further explanation.
4. Some answers confused the role(s) of objects and classes in OOA & OOD, likewise partitions and layers in U.M.L., and also concrete and abstract syntax (of an application).

Section B

More than 2/3 of the students have chosen B1, but most of them didn’t do well. Students who chose B2 did better. The average score for Section B is around 50% (marked out of 20). My observations are:

- Students in general don't have a good grasp of UML notations and their associated concepts. For example, some students don't know the difference between a System Sequence Diagram (SSD) and a Design Sequence Diagram (DSD), whereas others cannot distinguish between Activity Diagram and Sequence Diagram. About half of the students don't seem to understand the concepts of external actors and domain objects.

- Students in general find the algorithm hard and this is perhaps the reason why most of them have not chosen B2.

Overall, the performance of the exam reflects what they have learned from this course and how this course has taught them. The outcome is fully expected and I am not surprised at all.
Resit examination- student absent.

Comments:

20342 Software Engineering  Liping Zhao

1. I was astonished how few candidates did this question, and how poor the performance was. Many of those who attempted it could not write a simple Prolog program. Part b)---admittedly a little tricky---beat everyone. Part c) was well done in about half of the attempts. I had expected this question to be too easy.

2. Almost everyone attempted this question, with parts a)--d) well done. Only a few candidates scored highly on e) and f). Generally, performance was reasonable.

3. Almost everyone attempted this question, and performance was good. Nearly everyone could write the syntactic structure of simple sentences (with a few minor mistakes), and understood movement. Writing the DCG was harder, but even this was not bad. For that part of the question I gave up to 4 marks for getting the basic DCG structure right, up to 4 marks for handling movement and de-inflection, and up to 4 marks for technical accuracy.

4. Almost everyone attempted this question, and performance was embarrassingly good. Nearly every candidate produced perfect lambda-calculus computations. (I took off the occasional mark for failing to show working.) Particularly encouraging were the last parts of the question, which were definitely not a matter of just parroting what was done in class. The three or four students who got the very last part deserve an anonymous pat on the back.

Comments:

20442 Symbolic AI  Ian Pratt-Hartmann

1. I was astonished how few candidates did this question, and how poor the performance was. Many of those who attempted it could not write a simple Prolog program. Part b)---admittedly a little tricky---beat everyone. Part c) was well done in about half of the attempts. I had expected this question to be too easy.

2. Almost everyone attempted this question, with parts a)--d) well done. Only a few candidates scored highly on e) and f). Generally, performance was reasonable.

3. Almost everyone attempted this question, and performance was good. Nearly everyone could write the syntactic structure of simple sentences (with a few minor mistakes), and understood movement. Writing the DCG was harder, but even this was not bad. For that part of the question I gave up to 4 marks for getting the basic DCG structure right, up to 4 marks for handling movement and de-inflection, and up to 4 marks for technical accuracy.

4. Almost everyone attempted this question, and performance was embarrassingly good. Nearly every candidate produced perfect lambda-calculus computations. (I took off the occasional mark for failing to show working.) Particularly encouraging were the last parts of the question, which were definitely not a matter of just parroting what was done in class. The three or four students who got the very last part deserve an anonymous pat on the back.

Comments:

20512 Algorithms and Imperative Programming  David Rydeheard
Josh Knowles
Milan Mihajlovic