COMPUTER SCIENCE

Overall grade boundaries

Higher level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 14	15 - 28	29 - 39	40 - 51	52 - 63	64 - 75	76 - 100

Standard level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 15	16 - 31	32 - 41	42 - 52	53 - 63	64 - 74	75 - 100

Program dossier

Component grade boundaries

Higher level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 5	6 - 10	11 - 17	18 - 25	26 - 33	34 - 41	42 - 50

Standard level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 6	7 - 13	14 - 20	21 - 27	28 - 35	36 - 42	43 - 50

The range and suitability of the work submitted

The selection of problems and the presentation of dossiers in this session were generally good or very good. There were still a few unsuitable choices and most of these were games or game-related and of the “closed” type. Students who selected an open-ended problem generally scored better. Open-ended topics are those that can be easily simplified if they prove too challenging or equally easily expanded if time permits.

Having said that, an even better strategy is to make sure the problem is of suitable scope right at the start. Problems should obviously be capable of being completed by one person within a reasonable time frame. Objectives should be clearly and specifically stated and in most cases, four objectives may lead to a trivial problem, and eight or more to an overly complex one. Objectives should be stated in relation to the requirements of a real user from whom relevant data has been collected in some way. Objectives should be achievable and students should be able to clearly explain to the teacher how they will demonstrate their success.

The coversheets can also provide candidates with a useful checklist to ensure that they follow the required structure, including a table of contents. These cover sheets 5/PDCS can be issued at the start of work on the dossier.

Standard Level students generally managed to achieve 10 mastery factors, but a few students still had pointless additions in their programs simply to satisfy a mastery item. For example, a program with only one extra method that prints Strings to the console is not sufficient to show "mastery" of methods. Students should use methods as a matter of course, so there would normally be many methods in a dossier.

A number of students used arrays to store data, but never saved the data in a file. It is questionable whether this actually demonstrates mastery of arrays, as using arrays in this way leads to an unusable solution.

Some SL candidates are not being awarded mastery of flags/sentinels or use of additional libraries when it appears they could have been. Additional libraries are, for example, AWT or Awing GUI libraries, utilities such as StringTokenizer, ArrayList or LinkedList. Mastery of additional libraries should not, however, be claimed for using java.io for File Handling. Similarly use of ArrayList and associated methods for sorting do not demonstrate mastery of either arrays or sorting. Static arrays are to be used for mastery of this aspect.

Candidates often provide too little sample output of normal runs of their programs – this is essential for the moderator to confirm that mastery of aspects has been achieved, the program actually works as stated and to confirm the teacher's assessment in C1.

Mastery factors should be considered at the beginning of the process and at least 10, preferably 12, should be clearly identifiable before the teacher endorses the student's proposal. Students should be carefully guided in these choices and not simply left to do what they think is interesting to them. Students should be closely supervised at each stage of the dossier so that the teacher knows how each individual is progressing and are able to sign the 5/PDCS statement that the work is solely that of the student with confidence.

Teachers need to be sure that they understand what each of the mastery factors entail and should ask for advice from, for example, the Online Curriculum Centre. Many problems remain with the Higher Level mastery factors and the main ones are:

Polymorphism can be achieved technically by overriding methods of Java library interfaces such as `Object.toString()` and `ActionListener.actionPerformed(ActionEvent)`. This is considered trivial.

Supplying an empty constructor and then a second one to initialise data members of a class is also trivial in almost all cases. The candidate would have to justify their use of multiple constructors in some way to show that the solution benefits from such a design.

Parsing a text file or other data stream means more than simply using a method or two from the built in wrapper classes (eg `Integer.parseInt(String)` and `Double.parseDouble(String)`. This is just a conversion process from one data type to another. The concept involves taking a String of text, splitting or tokenizing it based on characters within the stream to produce data for subsequent processing.

Much more detail on individual mastery aspects can be found in the May 2006 Subject Report available on the Online Curriculum Centre.

Schools should be sure to see that accurate page numbers are provided when indicating where mastery has been achieved. Moderators are under no obligation to search for non-documented mastery claims; the onus is on the school to provide accurate information.

Candidate performance against each criterion

ANALYSIS

The Analysis section is the foundation upon which a good design is made and an effective solution created. Candidates who program first and then reverse engineer their analysis usually score poorly in this section and the rest of the dossier.

Criterion A1

Candidates who lack a real user and a real problem put themselves at an immediate disadvantage for most of the dossier. Many candidates still start by talking about the program they are going to write instead of the information problem they are investigating.

It is stretching credulity to claim that a large bank or hospital is using paper records and that they have asked a teenager to write an application for them. There are very many suitable small-scale problems in most international schools where the individual user is easily accessible and existing records can easily be seen.

Games problems continue to be difficult for most candidates who usually spend a little time on the programming and a lot more on GUIs, images and game play. Where a candidate is completing a game with images such as card decks and chess pieces, the source of these should be acknowledged.

The better examples set the scene and give some background so that the moderator can quickly assimilate the topic outline.

There should be evidence of data-collection. This means that actual data should be presented - actual numbers and strings from the problem domain. Many dossiers present explanations - e.g. "name and phone number" - but there should also be sample data presented - e.g. "Fred Thomas, 312-4567".

In an ideal case, a small manual system is being improved and existing documents from this system can be invaluable in providing sample data. Students of computer science should be adept at using scanners and digital cameras to capture and include relevant data – most other students are by now.

Criterion A2

Students should relate the goals to the analysis. This implies some sort of explanation why each goal is important. For example:

"Allow the user to search for a specific birth-date - so the boss can find out who has a birthday each day and then send a birthday card to the employee."

As stated above, around six goals or objectives of this type can create a project of a good scope.

Some candidates failed to discuss the limitations on memory usage, time response, error handling and system requirements. Often, usability was not discussed in any detail – “user friendliness” usually requires elaboration in the context of the problem.

This section is better done in bullet points rather than written in essay format which is both hard to read and to refer back to in later sections.

Criterion A3

Often an initial design for the prototype was not included or misunderstood. The initial design can be a very basic data-flow diagram or outline. The prototype should be something appropriate for discussion with the end user - preferably a user interface, but should also include sample data, not just menu items. Many students put an initial design in the section for A1 - it is preferable that this directly precedes the prototype in A3.

Some students presented a prototype but no initial design, this can be a very basic data-flow diagram or outline. The prototype should be something appropriate for discussion with the end user - preferably a user interface, but should also include sample data, not just menu items.

The discussion with the end user should be documented in some way, this was not always done or was rather trivial – *“The user liked what he saw”* sort of thing. It seems unlikely that the user has no questions at all when the programmer returns to them with a proposed solution to their problem.

DESIGN

It seems that the design section continues to cause students the most problems, and this is usually because a thorough analysis of the problem has not been completed.

Criterion B1

Data structures should be discussed and illustrated with sketches and sample data from the domain. The weakest candidates did badly in this section, failing to provide these items. Abstract diagrams of lists and trees are not useful in this section.

If the candidate decides to implement OOP features then this section is where some discussion of the need for polymorphism and or inheritance should appear.

Quite a few SL students described using arrays for storing data, but ignored using a data-file to store the data permanently. This is usually not an appropriate solution for a database-oriented problem - the most common type of problem. Lack of permanent storage makes an unrealistic and unusable solution, as well as making the testing in D1 difficult - dossiers without data files tend to present "lists" of data containing only 1 or 2 records. This creates problems for moderators as discussed further below.

Students should avoid long text explanations of their data structures, and should use diagrams to clarify and effectively communicate their thinking. This is often a suitable opportunity for peer review.

Criterion B2

The commonest problem here is cutting, pasting and perhaps doing some search/replace on the original code listing. This gains no marks and is a waste of candidates' valuable time.

Where candidates decide that the effort involved in producing a complete solution with pre and post conditions, parameters and return values is not worth the 2 marks allocated, they should describe most of the major algorithms – perhaps using a pseudo-code description with preliminary comments. This is a reasonable strategy for the less able candidate.

There is little point in attempting to design algorithms related to GUI implementation here, a better approach is to use input/output statements where relevant, and leave the detail until implementation time. Similar comments apply to Exception handling, if implemented.

Planning and describing algorithms ahead of time makes it easier to write a functional program without wasting time.

Criterion B3

This should also be a design section and not a description of the classes used in the solution. Bearing in mind that candidates may depart from the original design, a relatively simple set of modules making clear the connections to algorithms and data structures will be sufficient. If in doubt, a diagram is a simpler and better approach for most candidates as compared to a text-based one.

Some students did not include links to data-structures and algorithms.

THE PROGRAM

Criterion C1

Program listings are generally good with close attention being paid to good style. Students should ensure they include comments in the source code. Some Standard Level students wrote the entire program in a single method - this does not get any mastery marks for use of algorithms, and probably makes it more difficult to write the program. They would benefit from breaking up the program into methods

To ease the process of the moderator in confirming teacher's awards for mastery, Classes should start on a new page or have a clear separation from preceding Classes and should have a description at the beginning. A list of Classes with page numbers at the start of the code is extremely useful.

This will also help in the frequent cases where the candidate has indicated the page numbers for mastery aspects but they are incorrect and the teacher has apparently not checked them.

Criteria C2 and C3

The usability and error-handling sections are much improved this session and most candidates are either providing full descriptions with examples here or making a table of references to screen shots and/or code listings as appropriate.

In the usability section candidates who explicitly referenced the usability criteria of A2 scored higher marks than those who did not. This also illustrates the need to have specific rather than general criteria for usability. For example, *“my program will be user friendly”* is harder to prove whereas *“the navigation buttons will be in the same place on every screen”* is easier to achieve and demonstrate.

Criterion C4

A key problem in this section is the claim that a program worked well and performed as expected when none, few or unconvincing screenshots are presented in D4.

Unconvincing screenshots are, for example, claiming that a search/sort has been implemented or records can be added and deleted from a file/list/array but only ever showing the same single record in multiple screen shots.

Normally the teacher would be expected to run the program with the candidate and confirm that the output presented in D1 can be produced as claimed. Therefore if there is little output in D1 the teacher should encourage candidates to present more before awarding the maximum mark here.

A sound approach which makes the student think about producing an effective D1 is to provide a description of how the program fulfilled each goal, including reference hard-copy output that shows that the program actually functioned as expected.

DOCUMENTATION

This section should not be neglected by candidates who have completed a workable or partly workable solution as many claims of achievement in criteria and mastery aspects require proof here.

It is strongly suggested that where a candidate is unable to compile code to produce screenshots, teacher assistance is given to enable compilation and the production of these. Of course, the teacher must then ask the candidate to make clear which code has been modified by the teacher and no award should be given for those particular sections of code.

Criterion D1

As recommended previously, more attention should be paid to demonstrating that the solution works with normal test runs and normal data. Ten pages of dialog boxes showing, eg, wrong password entry or failed range checks, followed by one page of one record being added to an array makes it very difficult to award appropriate criteria for C1, D1 and the mastery aspects claimed, many of which require evidence to be produced.

Some candidates produce hundreds of pages for this section. Please don’t. It would be much better to carefully pick screenshots that demonstrate that each criterion in A2 has been achieved and that all claimed mastery aspects are working.

Testing of invalid output should not be neglected but probably should be about 33% of the total screenshots produced rather than 90%. It is better to concentrate on what the candidate has achieved.

Criterion D2

Evaluations are generally improving and often follow a structured approach addressing the criteria for A2 under the effectiveness section.

However, many students simply answer the questions listed in the guide, but do not really show evidence that they have thought critically about their program's performance. Many students listed web-functionality as a possible alternate design. In some cases this was realistic, in others it was pointless. The suggestions for improvements and alternative design should be sensible.

Criterion D3

There are few problems with the assessment of this section although some candidates find it a burden. Copying screen shots and giving a brief outline of the data required to be entered is a useful approach. Installation instructions are required for the highest award.

Students in several schools wrote lengthy instructions about how to install and start the program, but nothing about how to use it. This brings into question what "running the program" means, but in future, students should be told to include instructions about how to use the program, as that is what the users really need.

Recommendations for the teaching of future candidates

Please read the subject guide as to the format and other requirements of the program dossier. Some dossiers did not follow the mandatory format, did not use the Java language or did not include page numbers and tables of contents. All of these make moderation difficult and subject candidates to a high risk of losing marks. A suitable problem for the program dossier is one which:

- Includes an intended user who is a real and approachable person
- Involves something the candidate understands
- Has been carefully assessed by the teacher
- Has the potential to satisfy sufficient HL mastery aspects
- Has been researched and analysed

One strategy for achieving these aims is to have the candidate present their problems to their peers for comment. This can make them think carefully before starting work and gives the teacher an opportunity to assess the appropriateness of the task right at the start where many crucial decisions are made. At this time, candidates could also be given a copy of form 5/PDCS as a checklist so that they know exactly what they will be required to produce, and can work on report structure as they go. Some dossiers looked very hurriedly produced as if the candidate suddenly realised the need for page numbers, tables of contents and so on at the end of the process. The essence of the program dossier is adequate planning and following the cycle implied by the stages analysis, design, implementation and documentation. Teachers should be sure to complete the paperwork fully and completely so as to avoid delay in the awarding of grades to their candidates.

Higher level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 15	16 - 30	31 - 41	42 - 51	52 - 60	61 - 70	71 - 100

The areas of the program that proved difficult for candidates

An ongoing problem is the simple fact that student do not read the questions carefully, and provide general answers that often do not answer the question.

Q17 required students to appreciate the fundamental issue of processing data in a list, and while on the whole this question was done reasonably well, many students did not seem to understand the difference between a dynamic data structure and a static one.

In Q19 (a) the role of the program counter was not well understood. The program counter holds the address of the next instruction to execute. Also, the role of a stack in evaluating mathematical expression was poorly answered.

Q21 indicated that a number of candidates had difficulty constructing algorithms. Algorithm development is central to computer science and will continue to be assessed.

Q22 posed a number of problems to candidates. Possibly as this was the last question, students may have had to rush their answers. Knowledge of cylinders was limited and reference to parallel rather than serial access to data not well understood.

The levels of knowledge, understanding and skill demonstrated

Students on the whole showed a good overall knowledge. It was pleasing to see good knowledge shown on aspects of the course, such as: boolean logic, recursion, number representation.

The strengths and weaknesses of candidates in the treatment of individual questions

1 Feasibility report

This was in general well answered with only a few students concentrating low-level details about programs.

2 Systems maintenance

This question was answered well.

3 Test data
Some students concentrated only on invalid data rather than normal, invalid and extreme data.

4 Interpreter vs Compiler
A number of candidates did not seem to be aware of the fundamental translation process of converting higher-level code to lower-level code that can be executed.

5 Direct access
Most students indicated clearly that access could be gained without reference to the other members in the file or list. However, a number of students confused the concepts record in a file and data element of a list.

6 Touch screen application
This was well answered.

7 Hexadecimal and binary conversions
Students demonstrated a sound ability to manipulate binary and hex expressions.

8 Underflow
This posed some difficulty for a number of students. Underflow results when a number is too small to be represented and typically defaults to 0.

9 Boolean expression from circuit diagram
This question showed that most students were able to derive a Boolean expression and simplify it, although many students did not simplify AB + B to B.

10 Functions of registers within the CPU
This question caused a reasonable level of difficulty. The Instruction register holds the currently executing instruction. The accumulator stores the temporary results of arithmetic of logic calculations in the ALU.

11 Recursive algorithm
Question 11 (a) was done well with many students noting that a recursive algorithm calls itself and must have a stop condition. Part (b), the trace, was also well answered.

12 Real-time processing in a nuclear power station
This question was answered well.

13 Decimal to hexadecimal and binary conversions
Question 13 showed that some students lacked the skill to apply two’s complement representation.

14 Defragmentation software
This showed that students were aware that data blocks in file can become scattered and the defragging brings these together contiguously and hence decreasing access time. However a number also indicated that more disk space would be made available, which is not the case.

15 Polymorphism

Question 15 showed a pleasing increase in student knowledge. The most common answers indicated the use of sample method names which were selected via pattern matching on the parameters as implemented via multiple object constructors.

16 Big O of a quicksort

Question 16 was answered well.

17 An ordered list of names in main memory

(a) Many students did not make reference to the role of pointers, but were aware of the fact that insertion and deletion would be easier to achieve than using an array.

(b) Many students used diagrams and showed a good level of understanding of the basic role of the pointer (reference in Java) manipulation to insert the new bode. At a detail level students need to be aware not to overwrite pointers.

(c) Many students indicated the use of a binary tree, some indicating an array, which in general would not be suitable for a high level of addition or deletion because the list would need re-sorting, which is not the case with a tree.

18 Car part company with a file of stock records

(a) Many students indicated faster access, but few then noted that the records would not need to be kept in order.

(b) In general done well.

(c) Students focused on collisions and did not often list other concerns e.g. faster calculation or hashing space size.

(d) (ii) Most students indicated a collision but a number did not address how this could be addressed via overflow, chaining or probing. Very many students thought that an error of some kind would occur and that the record could not be stored at all.

19 Stacks and sub-program calls

(a) The program counter holds the address of the next instruction. Many students showed that a push placed the return address on the stack and that a pop restored the state but they did not incorporate appropriate reference to the fact the program counter is modified by the pop operation.

(b) Many students did not indicate that a stack is used and hence post fix allows the evaluation to be performed in a one pass manner.

(c) Well done.

(d) Some students struggled with this question and some diagrams did not resemble a stack at all. The most common error was to have the stack elements in reverse (ie. A at the top).

20 Small company LAN

(a) Students showed a sound level of understanding of the initial communication between two devices before transmission and gave simple examples.

(b) Done well.

(c) Done well, but many did not refer to the fact that a set of rules are used to guide the action of the firewall.

(d) Students showed a good level of understanding of how to safeguard data from unlawful access. The most common suggestions were login and password or encryption that would limit the success of any access gained.

21 Logic gates and associated algorithm construction

This question divided the candidates into two groups: those that could and those that could not.

(a) Done reasonably well, but a large number of students seem to be not able to construct a truth table or recognise exclusive or (XOR)

(b) Students provided a range of algorithms that worked. The most common error was to not exclude *A=0* and *B=0* returning true. A common answer was:

If (a == b) return true

Which is XOR as it allows *0,0* and *1,1* as conditions to return true. The correct condition is if *(a==1 and b==1)* return false, otherwise return true.

(c) Students provided a range of responses and it was pleasing to see the level of expertise displayed. A number enumerated all possibilities in a long if statement, which was acceptable. The key was to see that if one false outcome occurred equality could not exist and the algorithm should terminate i.e. return false.

A number of students used the test *if(logic1(a,b) == logic2(a, b) return true*. However, this does not terminate the looping process because the first equality would end the checking.

The key test is to return false for the check, *if (not (logic1(a,b) == logic2(a,b))*

(d) Many students could reduce the expression and showed an appreciation of Boolean logic and the application to laws.

22 Functions of the operating system and format of a hard disc

(a) Done reasonably well.

(b) Track and sector is well understood but the notion of cylinder was not as well appreciated, which showed up in part (c)

(c) Many students attempted a technical explanation that was not appropriate. Cylinders allow parallel reading and writing, hence more data can be handled in one cycle.

(d) A blocking approach simply reduces the number of read or write access cycles, few students could give this as an answer.

The type of assistance and guidance the teachers should provide for future candidates

Teachers need to continue to educate students in the art of question reading. Also, the number of marks allocated to a question indicates the number of points to be expected in the answer. Algorithms need to be central to exam preparation and students should practice this type of question under examination conditions.

Higher level paper two

Note: There was an issue with the case study question, in that there were only 30 marks assigned to it, not 40 as required. This resulted in the paper being marked out of 90, and the scaling factor was adjusted.

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 15	16 - 31	32 - 38	39 - 47	48 - 56	57 - 65	66 - 90

The areas of the program that proved difficult for candidates

The standard of work varied. Although there were some excellent scripts, basic computer knowledge and familiarity with terms was one of the weakest areas.

The levels of knowledge, understanding and skill demonstrated

Candidates find it hard to focus on the discussion type questions, writing far more than the mark allocation would suggest without addressing the significant points.

Most candidates had enough knowledge to attempt all of the questions. Often a candidate's knowledge and understanding of a topic was evident but the discussion was insufficient to earn full marks.

In every question there were some excellent responses showing that the syllabus coverage in many schools is very good.

The strengths and weaknesses of candidates in the treatment of individual questions

1 Linked list and recursive algorithms

Most candidates who answered this question made a good trace of the code. Some candidates did not clearly label the head and the links between the nodes. Some candidates showed the list in reverse order. Most candidates were able to state the efficiency of adding nodes to the list and efficiency of searching through the list.

Most candidates did make a reasonable attempt to trace the recursive method and some answered this perfectly.

Answers to constructing the method that returns true if the value is on the list and false otherwise ranged from poor to excellent.

Explanation of how a binary tree could be searched was generally well done.

2 Binary arithmetic, fixed and floating pint representations

This was the best-answered question. Most candidates were able to present numbers in binary format, and convert numbers from decimal to binary. Some candidates were not able to calculate the value of the number represented by the mantissa and exponent.

Most candidates clearly outlined one example of computer application that uses floating-point values, and stated why errors in representation caused problems.

3 Methods of file organisation for a telephone directory

This question was really well answered by a few candidates, but most did write good answers to some parts of it. Candidates who scored less marks were the ones who suggested an inappropriate alternative in Part (b). Some were not even sequential and many used a binary tree or linked list. A similar problem followed from Part (f) to Part (g).

The candidates did not appear to know very much about the methods of file organization. Many students used terminology incorrectly, for example term file is not interchangeable with record. File organization is a very weak area for candidates, although it is an extensive topic in the Higher Level syllabus.

4 Case study question – computers and disability

Candidates did fairly well on the question as whole, which you would expect given that they have time to prepare for it in advance of the examination.

Parts (a) and (b) were the best answered. Some candidates considered disability issues in relation to output devices rather than the input devices specified in the questions.

Most candidates could outline a suitable data structure that could be used to store abbreviations and the longer phrase; and were able to suggest a method of accessing the abbreviation.

Part(c), (d) and (e) were related to data structures and so were well answered by many candidates, who earned almost full marks.

Most candidates answered Parts (f) and (g) well.

In Part (h) many students simply reiterated the general issue but did not provide social implications related to the increased dependency of society on computer networks. Also many answers strayed from the main point, which was **lack** of access to networks.

The type of assistance and guidance the teachers should provide for future candidates

Candidates, especially weaker ones, should spend more time tracing and constructing algorithms. They need to develop their confidence in understanding and writing these under examination conditions.

The more examination practice the candidates get the better. Old exam questions could be integrated into regular teaching-learning process. This will allow reinforcement of subject matter and make them familiar with how the questions are structured and asked in the examination.

More time could be spent using the case study as a teaching tool in different contexts and not only in the given scenario.

Standard level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 12	13 – 24	25 - 29	30 - 36	37 - 43	44 - 50	51 - 70

The areas of the program that proved difficult for candidates

It is clear that some schools do not prepare their candidates well, if at all, for the theory papers. Many candidates had little or no knowledge of topics on the paper. In some cases candidates were unable to complete any questions except those on programming and number topics.

The lowest mark was zero and it is truly alarming to think that a school would not even direct candidates to a book or website on computing before subjecting them to such an ordeal.

The levels of knowledge, understanding and skill demonstrated

While some candidates demonstrated a generally high level of knowledge and understanding, it was evident that many were quite unprepared for this paper.

Candidates tended to have poorer knowledge of computer hardware as compared to the areas of programming, logic and end-user experiences.

Many answers suggested that candidates could have answered the question had they thought more about what was being asked, how the marks could be earned and by replying with greater precision. It is evident that many students did not read the questions and were unable to use terminology correctly.

Some candidates tried multiple answers to questions and this is a strategy to be avoided. Many candidates appeared to have difficulty in communicating to the examiner and writing skills should be improved by past paper practice in some cases.

The strengths and weaknesses of candidates in the treatment of individual questions

1 Feasibility Report

There were many components that could have been identified and many candidates could identify at least two. A common error was to include items that would have been produced in a subsequent analysis of the problem, eg fact-finding techniques or hardware requirements. Less common were answers referring to systems already in place.

2 Interpreter vs compiler

Most candidates were able to answer this question adequately. While Java and modern IDEs have the potential to confuse this issue (by compiling to byte codes which are then interpreted at run time, with run time errors being subsequently indicated by the IDE) most candidates still recognise the essential differences between these programs.

3 Register

Very many candidates confused a register with the process of “registering” applications in an operating system. Other candidates were able to describe the function of registers in relation to the following question on the machine instruction cycle. The best candidates were able to explain the relationship between registers, words and buses in the CPU.

4 Instruction cycle

Some candidates used the term data instead of instruction here – fetch data, decode data etcetera. Others confused this cycle with the Input-Process-Output model of computer systems. Some candidates stated the steps rather than outlined them. Care should be taken over action verbs.

5 Router

Many candidates identified a router as a device that connects a network to the Internet. While it may be the case that a modern device combines the functions of a broadband modem and a wireless router, the difference between the two should be known.

6 Hexadecimal binary conversions

This question was very well answered on the whole and caused problems mainly to those who had clearly never been taught this topic. There were few “half right” answers.

7 Analogue and digital data

Many candidates could correctly distinguish analog(ue) and binary data and understood the need for conversion or could give sound examples of analog data requiring conversion. The need for sampling or other methods of conversion was rarely mentioned, however.

8 Data entry errors

This question posed no problem to the majority of candidates.

9 Data compression utility

This question posed no problem to the majority of candidates.

10 Arrays

Many candidates demonstrated familiarity with arrays and indexes and this question was mostly well answered.

11 Data integrity

The most common error was to give answers relating to encryption, and candidates do not always demonstrate the difference between integrity and security of data. Some candidates confused check digits with check sums. Some answers referred to sending data twice to compare files and this may result from a casual reading of the question, which refers to “during the transmission” of data.

12 Hard disks

While Standard Level candidates are not expected to know the details of hard disk formatting into sectors, tracks and cylinders any discussion of secondary memory and type of access should bring out some of the features, which allow direct access. For example, movement of the head to any part of the disk, the ability to read a block without traversing other parts of the medium and the inclusion of an address table within direct access media of all types.

13 Tracing an algorithm to insert spaces between strings

This question was understood and answered well by most candidates. Some candidates made assumptions rather than tracing the code effectively, for example, assuming the asterisks meant that it was some kind of password entry system. Quite a few candidates also understood that it could be used to line up columns of text in a table.

14 About a business with stock files in various cities

For part (a) many candidates gave examples of verification rather than validation. Most candidates could gain the marks for parts (b) and (c). The question about updating files was also well answered for the most part while the final section was answered well by about half the candidates attempting it. There were some suggestions that the network not be used at all, eg by faxing or telephoning the branches; while true this does not seem an appropriate response to the problem.

15 Doctor’s surgery/office and networks

Again, this question seemed largely straightforward for many candidates. Although part (b) was generally poorly answered, many candidates did not know what the hardware component was, or if they did were content to name it (gateway/router/modem) without explaining its functions. Almost all candidates could answer the questions in (c) and (d) without difficulty as this topic has been well explored in recent papers.

The anglo-centric term surgery was a potential source of confusion for some candidates and due allowance was made in marking the question, although it did not affect the perceived situation greatly. Answers relating to surgical operations were equally well accepted when discussing benefits/risks to patients.

16 DVD recorder and microprocessor

There was some evidence that a few candidates were unfamiliar with the operation of a DVD recorder and some were evidently misled by being given the answer to (a) in the first part of the question. Perhaps they thought it was a trick question (something which the examiners avoid).

Most candidates could identify the required inputs although again, the handheld control featured in many responses (incorrectly).

Many candidates could recognise this as a real-time application and could identify the key characteristics of real-time processing, even those candidates who did not answer (a) to (d) well could often gain these marks.

The type of assistance and guidance the teachers should provide for future candidates

Teachers should give more emphasis to the non-programming sections of the curriculum. While programming skill is necessary for successful completion of the dossier it should be recognised that hardware, architecture and networking are also significant areas that need to be studied.

Terminology is a not particularly exciting part of the course but candidates have little chance of success unless they know it well. Therefore interesting and perhaps competitive methods should be used to encourage candidates to memorise computer jargon.

Teachers must also encourage candidates to attempt many test papers before the examination and to reinforce sound examination techniques, such as reading the question, observing the action verbs used and responding in accordance with the allocated marks.

As mentioned above, candidates should be discouraged from giving multiple answers to a question; only the first answer will be marked even if the correct answer appears later on in the response.

Unqualified answers are only acceptable when the action verb “state” is being used; slower, better, smaller and so on must be qualified or compared to earn marks, particularly when an explanation or description is being sought.

Standard level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 11	12 – 23	24 – 30	31 - 36	37 - 41	42 - 47	48 - 70

The areas of the program that proved difficult for candidates

The structure of the paper is made up of three questions, two of which contained a large amount of algorithm content with the third dealing with the Case Study. The paper produced a wide range of marks, with all questions proving accessible to the better students. Most students tended to do well on the Case Study question.

The lower marks were not so much as a result of the algorithm questions being particularly difficult, but more as a result of algorithms and logical problem solving being dealt with differently in different schools, as there was a marked difference between the results of schools. There are still centres that fail to prepare their students adequately for what is always a significant part of this paper. Conversely, the students from those schools who do prepare their students well, scored well, as the algorithms were relatively straightforward.

With regard to the Case Study, questions that begin with action verbs such as “explain”, “discuss”, “compare”, continue to prove difficult for many students, who tend to offer no more than a descriptive response.

The levels of knowledge, understanding and skill demonstrated

As indicated above, the areas that proved difficult for some students were the same areas in which other students (from different schools) showed considerable skill and understanding. Students from schools who prepare them to think logically through the construction of algorithms will always pick up a reasonable number of marks on this paper. Common processes such as searching, sorting, inspecting array values, returning values to the calling method will always come up. Many students, consequently, were able to construct the algorithms that were both logically correct and in many cases elegantly written. The ability to complete these algorithm questions quickly also allows these students more time to spend on the Case Study question.

The strengths and weaknesses of candidates in the treatment of individual questions

Question 1

Arrays are a data structure regularly tested, and access to their contents using loops (for 1-D arrays) and nested loops (for 2-D arrays) often forms parts of questions at both SL and HL. Students, however, still make mistakes with the indexes, in this case many mistaking the “Floor” labels for another column of data (in part (a)).

The methods requested in (b) and (c) were quite straightforward, but is clear some schools are still giving their candidates inadequate preparation in handling nested loops, as they struggled with the terminating conditions. The fact that the initial index of an array is 0 is by now well established, with the maximum index being the (array length -1).

Again, the “average” question in (d) was not complicated, but some students disappointedly avoided the loops and instead included lines of additions. When “appropriate output messages” are requested, the students are normally being tested on their ability to include variables within the message (in this case the correct floor).

1 (e) asked why an array structure was an appropriate structure to use. This proved more difficult to answer than actually writing the previous algorithms, with only the better students offering correct answers, most of which dealt with the means of accessing the data through indexes and loops.

Question 2

This introduced some elements of Object Oriented Programming, which includes concepts that are difficult for most students and usually only dealt with at HL. But SL students should be able to define “constructors” (parts (a) and (b)). The question tended to differentiate between the more knowledgeable programmers and the rest. Many are not clear exactly what creating a new object of a particular class entails. A lot of students are familiar with the main method consisting of no more than a call to the constructor - e.g. new Numbers() – with the constructor then calling the various methods present in the (1-class) program. This has unfortunately led them to believe that the role of a constructor is to run a program (and not to create a new object).

Parts (b) and (c) were again quite straightforward algorithms. Some made the mistake in (b) – finding the maximum – of not using a temporary variable to hold the maximum, but to instead comparing successive values of the array. It can be solved successfully this way, but most students would have encountered an “array out of bounds error” by requesting an index 1 more than the maximum.

Most identified the search in (c) as being sequential and offered a “binary search” as a better alternative in (d).

Having been introduced to the two methods in (b) and (c), most students made a reasonable attempt to include them in the main method in (e). Only the more OOP-wise students realized that you had to reference the methods to a particular object (in this case “array”) with the use of dot notation.

Question 3

Most students gained reasonable marks on the Case Study question. Students from schools who had devoted time to the Case Study clearly fared better with fewer “verbatim” answers taken straight from the Case Study itself.

Questions that use the action verbs “Discuss”, “Suggest”, Explain”, require more than a brief descriptive answer. Students who answered in that way would normally not gain full marks for each point.

For example, in (a) (i) , stating that the person would have difficulty in using a keyboard and mouse would have gained 1 mark, but following this up with an example of an alternative design (larger keys, keys grouped close together head mice etc) would have gained the second. Similarly suggesting an application of biometric control in (b) required both a description of the application and a means of achieving this.

Part (f) ended the paper with a reasonably difficult question. The question asked for “implications of the increased **dependency** of people on computers”. Only the best students answered this the way it was asked, with the majority dealing with the “increased **use** of computers” which is not the same.

The type of assistance and guidance the teachers should provide for future candidates

Problem solving by use of algorithms is a fundamental part of the course and must form a significant part of the teaching of this subject. Schools should also encourage their students to look carefully at the format of each question, including those that appear in the Case Study, to ascertain exactly what is being asked, and to not simply write a descriptive answer that may only partly answer the question.