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 - 14	15 - 30	31 - 42	43 - 53	54 - 64	65 - 76	77 - 100

Higher level Program dossier

Component grade boundaries

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

The range and suitability of the work submitted

The selection of problems at Higher Level is necessarily more difficult and care should be taken that all required mastery aspects are planned for at an early stage in the process. Almost all students chose a standard database oriented problem. Only a very few tried to do something novel and. sadly, these students generally encountered problems. However, the extended mastery aspects for 2010 should allow more flexibility here.

Most students understood the significance of the assessment criteria and modelled their work to address those criteria successfully. It appeared that many teachers had shown their students a model dossier and encouraged their students to do something similar. However, caution should be exercised in taking an overly templated approach which may end up with the candidate rarely thinking through the issues for themselves and can also lead to the appearance of collaboration or plagiarism. This is a fine line to tread.

Basic OOP concepts remain poorly understood in many cases and are often used in a trivial manner. For example, the mere use of the `extends` keyword does not satisfy mastery of inheritance. Where a candidate is using inheritance it should be planned and the candidate

must develop all of the classes involved. The only exception might be when candidates expand significantly on a pre-written library class and can explain why they took such an approach.

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

Recursion continues to be used and claimed in situations where a simple iterative loop would serve as well, if not better. This is also considered trivial.

Appending to a RandomAccessFile instance by using the seek method as in:

```
myFile.seek(myFile.length())
```

is also a trivial example of adding data to an instance of the RandomAccessFile class.

More information on individual mastery aspects can be found in the table at the end of the recommendations for the teaching of future candidates as well as in the May 2006 Subject Report available on the Online Curriculum Centre.

Candidate performance against each criterion

Generally students achieved similar marks across all the criteria. That is, weak students had low marks in many criteria, while strong students had high marks in most criteria.

The criteria that were generally well done were: A3, C1, C2, C3, D2 and D3. In the other criteria there were a variety of problems.

Criterion A1

In A1, many students start off with a general, non-specific overview of the problem, and then jump into a discussion of their solution. There should be a gradual expansion of the details of the problem that leads logically to a **proposed** solution. Often the background is neglected, making the problem unclear to the reader (as for SL dossiers).

Criterion A2

Many students wrote a sensible list of goals. In many cases, the list of goals was so long that it would be virtually impossible to provide enough sample output to demonstrate the success of the program. Also, many of the goals are pointlessly detailed or involve uninteresting technical details, e.g. "the program should store the data in a RandomAccessFile using 100 Bytes per record." While the criteria should be specific and measurable, technical detail is not required.

Criterion A3

Some students omit or misunderstand the requirement for an initial design, which can be a simple module diagram arising naturally out of sections A1 and A2. Evidence of user feedback is often patchy or unconvincing.

Criterion B1

The data structures should be explained in relation to the requirements of the proposed solution. Many students outline a generic linked-list but without sample data or explanation of how the linked-list would be used. Thus, this documentation provided no information that would help the reader understand the overall design of the solution.

Criterion B2

This requirement will be considerably reduced for 2010 onwards. This ought to reduce the need for students to summarize (or even copy) the finished Java code. This section is only rarely done well.

Criterion B3

Often students simply present a description of the classes they have written already and many teachers believe this to be worth 4 marks, unfortunately. A simple module diagram goes a long way towards meeting this criterion, especially if connections to data structures and algorithms are included.

Criterion C1

Code listings are generally good. Candidates still do not always mark out the code that has been generated by an IDE from code they have written themselves.

Criterion C2 and C3

Note that the Usability section will not be required from May 2010.

The error-handling section should refer to code examples either quoting code or referring to specific methods and lines precisely.

Criterion C4

There is no separate documentation required for this section.

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.

Criterion D1

It is important that Higher Levels candidates use this section to prove the successful mastery of claimed aspects, where appropriate. Teachers should strive to ensure that all candidates provide at least one full sample run of the solution although, as per the guide, more are really required. Less emphasis need be placed on invalid data, however.

Criterion D2

As for Standard Level, this section is not always thoroughly done. Efficiency should be discussed at this level and suggestions for future modifications should be sensible and realistic – i.e. within the capabilities of the proposer.

Criterion D3

This section will no longer be required from 2010 onwards.

Recommendations for the Teaching of Future Candidates

The problems in A1, A2, B1 and B2 were often the result of students writing a program first.

Teachers should emphasize the importance of sample data for both the ANALYSIS and the TESTING of a computerized system. Although a rigorous set of test-cases (test strategy) is no longer required, students still need to understand that the users care a lot more about the data than the software. Correct handling of data is the primary goal of most of their programs.

Teachers should insist that students do the analysis and at least part of the design before they start programming. Then the teachers need to give ample guidance when the students are on the wrong track. Teachers need to recognize when a student is thinking about a solution that will be inappropriate (either to simple or too complex) and warn the student and help them correct their thinking in the early stages of the project. This saves time and effort and greatly increases the student's chance of success.

Almost all students have adopted the safe strategy of choosing a data-base oriented problem implemented as a console-based program. It is hoped that the changes in the IA assessment criteria – especially at HL – will encourage students to be a bit more adventurous and we will have more students attempting other types of problems.

The information on individual mastery aspects in the table below should be read in conjunction with the May 2006 Subject Report available on the Online Curriculum Centre.

HL MASTERY ASPECT	WHAT IS ACCEPTABLE	WHAT NOT TO DO
Adding data to an instance of the RandomAccessFile class	The seek method must be used to move the file pointer to a specified byte in the file before writing.	• seeking to the start of file – seek(0) – and writing a stream of data to the file. • seeking to the end of the file seek(file.length) and appending data to the file.
Deleting data from an instance of the RandomAccessFile class	the seek method must be used to move the file pointer to a specified byte in the file before reading.	Seeking to the start of the file (seek(0) and then reading the entire file is a trivial use.
Searching for specified data in an instance of the Random AccessFile class	No clarification necessary	No clarification necessary
Recursion	No clarification necessary	• Trivial use of recursion is a use where an iterative solution would work equally well (eg counting nodes or searching in a linked list). • Recursion should not be claimed when using a standard sort algorithm (mergesort or quicksort) that has not been documented at the design stage.
Merging of two or more sorted data structures	Merging requires two data sets which are already sorted.	The most common mistake here is to have a mergesort.
Polymorphism	No clarification necessary	Candidates may not use library code in their mastery claims and candidates have not written either of these superclasses.
Inheritance	The candidate must document and demonstrate	Cannot simply extend a built-in

	how the subclass uses the methods and data members of the parent Class.	or library class.
Encapsulation	No clarification necessary	No clarification necessary
Parsing a text file or other data stream	No clarification necessary	Using the built-in wrapper classes such as Integer and Double with their methods `parseInt(String)` and `parseDouble(String)` is considered a trivial use.
Implementing a hierarchical composite data structure	Examples can be found in the subject guide on page 65.	No clarification necessary
The use of any 5 standard level mastery factors	See table referring to awarding of SL mastery factors.	No clarification necessary
Up to four aspects can be awarded for the implementation of abstract data types (ADTs)	Where one or more candidates are using the same ADT the teacher must make a note regarding each candidate and the candidates must individually justify their use of the data structure – this will be different for different problems.	No clarification necessary
Use of additional libraries	The student must also write some code themselves that uses the library, and this code should be clearly identified as such.	Do not claim for libraries that are required for other masteries to work (such as `java.io.*` for file handling). Using classes created by the student does not count.
Inserting data into an ordered sequential file without reading the entire file into RAM	The student is expected to do more than simply store data	Students at HL should avoid the use of the SQL library with an external database or serialization.
Deleting data from a		

sequential file without reading the entire file into RAM		
Arrays of two or more dimensions	Therefore candidates should use standard arrays of primitives or classes with subscripts in [square brackets]	The java.io.ArrayList Class or various other vector-style implementations should be avoided as this aspect refers to traditional static arrays

Standard level Program dossier

Component grade boundaries

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 some unsuitable choices and these were either games, game-related or limited in some other way. Some projects could simply not achieve the required mastery aspects or there was no real user and this prevented the candidate being able to scope the problem sensibly.

A handful of dossiers were far too complex and typically this involves too many classes with too many instance variables which increases the students' workload without benefit.

Objectives should be clearly and specifically stated and in most cases 4 objectives may lead to a trivial problem and 8 or more to an overly complex one. Students should be able to clearly explain to the teacher how they will demonstrate their success and the teacher must give adequate guidance in this area. If a teacher is unsure because of inexperience with the programme than (s)he should ask for advice on the OCC.

Students who chose an end user who was close and accessible (such as a teacher or close relative) found it much easier to gather the required data for the analysis and goal-setting parts.

Standard Level students generally managed to achieve 10 mastery aspects 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.

Mastery aspects 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. Teachers need to be sure that they understand what each of the mastery aspects entail and should ask for advice from, for example, the Online Curriculum Centre, if necessary as well as using the information in the table after the recommendations for the teaching of future candidates.

There was some evidence that some teachers are not confident in this area and simply followed the candidate's advice (wrongly).

File i/o should involve writing and reading files (text files are fine) not only one of those. Arrays should be static types and library utilities such as ArrayLists and Vectors should not be used for this aspect.

Where simple "setter" and "getter" methods and constructors are used as part of an Object the same methods should not also be used to claim mastery of methods with parameters and methods with return values as such methods are usually trivial. In addition, some candidates appear to use the default public scope specifier in such a class which makes these methods superfluous anyway.

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.

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. It is virtually impossible to confirm mastery of arrays, file i/o sorting and searching etc when there is no hard copy given. The only recourse a moderator has is to try and interpret the potential success or otherwise from the code listing and few moderators have this kind of patience (nor are they expected to). It is the candidate's responsibility to demonstrate success.

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 should not write qualifications on this form such "the student submitted the dossier only on the due date and therefore I am unable to confirm that it is their own work".

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

Many candidates start by describing the program they are going to write (or probably have written) instead of the information problem they are investigating. Many students assume knowledge of the problem domain and should be encouraged to give a general introductory background.

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". Students who were able to provide this data almost always scored better overall on their dossier than those whose user was illusory or patently made up.

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.

A systematic analysis was rarely done. In a systematic analysis, the processes involved in producing the required outputs from the carefully identified inputs are described in some way. This could be descriptive or diagrammatic (user stories, use cases, data flow diagrams etc). Like all diagrams in the dossier these should be related to the problem at hand rather than generalized examples.

Criterion A2

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

> *"The list of cars should be searchable by colour or number of doors as users often have specific requirements when purchasing a used car"*

As stated above, around 6 goals or objectives of this type can create a project of a good scope. The goals could be put to the SMART test – at the very least they need to be Specific and Measurable so that the student knows when they have completed them (for C4) and is able to demonstrate them (D1) and discuss them (D3). (Achievable, Relevant and Time-constrained are also important but students will usually need teacher guidance in this respect as noted above).

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

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.

Some students simply include their screenshots from the solution here, prototyping is a useful transferable skill so this is a lost opportunity sadly.

DESIGN

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

Criterion B1

GUI objects such as text boxes, drop-down lists and the like are not generally considered relevant to this section. SL students need to focus mainly on the data types they have used, any data structures (primarily arrays and files) that they are planning to use. A description of a record style object should also be included if used.

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.

Students should avoid long text explanations of their data structures and use diagrams to clarify and effectively communicate their thinking. This is often a suitable opportunity for peer review. Students should be encouraged to keep their early design notes for this section.

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.

This will be less of a problem for 2010 onwards as detailed algorithm descriptions will no longer be required.

Criterion B3

A relatively simple set of modules making clear the connections to algorithms and data structures will be sufficient for most candidates. If in doubt, a diagram is a simpler and better approach for most candidates as compared to a text-based one.

THE PROGRAM

Criterion C1

A clear and consistent page numbering system should be used and teachers should double check that the mastery aspects are correctly referenced. Code listings are good in general although sometimes odd fonts and line-spacings make listings hard to read. A mono-spaced font is the best option.

Criterion C2 and C3

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 in section A2.

Note that the Usability section will not be required for May 2010.

The error-handling section should refer to code examples either quoting code or referring to specific methods and lines precisely.

Criterion C4

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. However, this is not a requirement of the dossier, just a suggestion that will help the student evaluate and reflect on the work they have done.

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.

Criterion D1

Candidates should carefully pick screenshots that demonstrate that each criterion in A2 has been achieved and that all claimed mastery aspects are working. The guide states that one run with valid data is rarely sufficient. More sample runs should be made with valid data than with invalid data. It is much better that the candidate emphasizes what has been achieved. Tables of proposed test data are not essential but candidates who did this approach generally scored better.

Testing of invalid output should not be neglected but probably should be about 30% of the total screenshots produced rather 90%.

Evidence that multiple records can be added to arrays and files is essential for the confirmation of the award of these mastery aspects.

Criterion D2

Students are often reluctant to mention the parts of their solution that did not work or work well, possibly they think examiners don't notice. Sometimes the focus was more on possible improvements at the expense of the actual solution.

Many candidates are justifiably proud of their achievements but this should not be the sole consideration in an evaluation.

Criterion D3

This section will no longer be required from 2010 onwards.

Recommendations for the teaching of future candidates

The largest single issue in this session was the lack of sample runs of valid data and candidates and teachers should bear in mind that this is pretty much the examiners only way of assessing what the candidate has actually achieved.

As indicated, a great deal of work needs to be done before the project, let alone the coding, is started. Teachers need to check early with students, the nature of the project and to see where mastery aspects are going to come from and that there are sufficient mastery aspects. They also need to make sure that the candidates understand the criteria. Where teachers themselves are unsure about mastery aspects they need to seek clarification at an early stage of the process.

One way to clarify the initial thinking is to ask the question "what would a solution to this problem have to be able to do?" to guide their way through the criteria for success. Having a real user is the best way to achieve success in the analysis A1 – A3.

In some cases, very large and complex problems were attempted. Even where these are successful, the candidate will have had to neglect some other aspect of their study. Keep the problem simple and don't feel that every user request needs to be addressed. Once again teachers have an important role to play in guiding candidates. A good SL dossier for the new 2010 criteria can probably be achieved in 40 pages.

The information on individual mastery aspects in the table below should be read in conjunction with the May 2006 Subject Report available on the Online Curriculum Centre.

SL MASTERY ASPECT	WHAT IS ACCEPTABLE	WHAT NOT TO DO
Arrays	Implement a standard array of Java primitive type e.g. int [] score = new int[10] or of type object e.g. objectName [] list = new objectName[10].	DO NOT USE ArrayList or any other list or collection such as HashMap, etc.
User defined Objects	Object use claimed must be unique to the problem solution and created by the programmer e.g. circle object with a radius.	DO NOT claim using built-in Java classes e.g. Swing classes. Certain classes such as Exception Handlers can be claimed provided the student adds a reasonable level of complexity and justifies its use.
Objects as data records	Typically an object with associated data and methods used to store an entity whose data forms a record in a file.	DO NOT use any built in Java classes.
Simple Selection	No clarification necessary	DO NOT claim any code not written by the student
Complex Selection	No clarification necessary	DO NOT claim any code not written by the student
Loops	No clarification necessary	DO NOT claim any code not written by the student
Nested Loops	No clarification necessary	DO NOT claim any code not written by the student
User-defined methods	No clarification necessary	DO NOT claim any code not written by the student
User-defined methods and parameters	No clarification necessary	DO NOT claim any code not written by the student
User defined methods with appropriate return values	No clarification necessary	DO NOT claim any code not written by the student
Sorting	Student written code to perform a bubble, selection	DO NOT claim using inbuilt Java sort methods associated with a

	or insertion sort. See teacher support material on the OCC for examples.	Java class e.g. Class Collections or Java.utils.Arrays.sory(arrayName)
Searching	Student written code to perform a linear search or bubble search. See teacher support material on the OCC for examples.	DO NOT claim using nay inbuilt Java search method associated with a Java class e.g. Class Arrays BinarySearch()
File I/O	Sequential File Read AND Write operations written by the student using BUfferredReader and PrintWriter as outlined in the Subject Guide or similar classes e.g. FileOutPutStream or DataInputStream. Object serialisation can be used at SL to read and write data to and from permanent storage but must be well explained and justified. Use of Java connections to external databases such as SQL is allowed for file I/O but not to claim other mastery of sort and search.	Avoid use of object serialisation with Java classes such as ArrayList or HashMaps if looking to claim other mastery factors relating to searching and sorting.
Use of additional Libraries	Can be claimed for any valid use of a Java class associated with solving the problem.	DO NOT claim for the use of classes that are not used to solve the problem.
Use of sentinel or flags	Sentinel can be used to indicate the end of a sequential file, end of data in an array or end of data when entering from the keyboard. A simple Flag can be implemented to terminate a bubble sort or linear search.	DO NOT claim for any sentinel or flag implemented in a Java class not written by the student.

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 - 62	63 - 72	73 - 100

The areas of the program that proved difficult for candidates

Many candidates performed reasonably well. Schools have to reiterate the fact that precise and specific answers are required of their candidates.

The interfacing problem (Question 18) has probably not been considered by most teachers/candidates. Only the very knowledgeable candidates correctly answered this question.

The levels of knowledge, understanding and skill demonstrated

There is a wide knowledge base of candidates appearing for this examination. The syllabus seems to be covered by most schools, the performance of many candidates hovered above the average. There were a few students who were either outstanding or very poor in their performance.

Many candidates write answers that are very brief and tend to neglect the amount of marks allocated to each question.

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

SECTION A

All of the questions in this section seem to be accessible to all the students and the majority showed a good coverage of the course.

Question 1

This was answered well by many students. A few candidates wrote “RAM and ROM are both stored in the computer”. They need to focus on the terminology more carefully.

Question 2

Many candidates did not find the question on operating system difficult. A few candidates gave very elaborative explanations on the functions of an OS, which were not expected.

Question 3

Question on 5-bit two's complement number representation was confidently answered by many students.

Question 4

Many good answers were written for advantages of "data compression" like less disk space (more data in reality); faster writing and reading and faster file transfer.

Question 5

Surprisingly many students wrote "compiler converts high level language to machine code" instead of "compiler converts source program written in high level language to machine code".

Question 6

Describing difference between a syntax and logical error was done with confidence.

Question 7

Surprisingly many students answered that user/defined methods are designed by end-user instead by programmer.

Question 8

Many good answers had points like "reduced development time"; "many programming teams are involved", "reduced errors".

Question 9

The question involving systems analysis and design was answered relatively well by all candidates.

Question 10

Well answered question.

Question 11

The Boolean expression was simplified successfully by a majority of candidates.

Question 12

BigO (the "o" stands for "order of") notation is concerned with what happens for very large values of N, therefore only the largest term in a polynomial is needed. Some average candidates just wrote vague answers to this question that did not warrant any mark.

Question 13

Most students scored well for defining the meaning of the term WAN (Part a) but explanation of one advantage of using packet switching in WAN (Part b) was given only by some students.

Question 14

Well answered question.

Question 15

Variables that exist only inside a block/method are called local variables - they have local scope. This was normally answered well by many students.

SECTION B

Question 16

Most candidates who attempted this question either did extremely well or very poorly. This seems to suggest that their programming skills and understanding was good and they could apply their knowledge to this question or lacked them.

Question 17

While answering the question involving the keyboard providing access to a computer room some of the students gave extensive explanation on how the stack operations ("push" and "pop") help in processing.

Question 18

The only real difficulty in this question was with Part c. The interfacing problem has not been considered by most candidates.

Question 19

Some of the smarter candidates did not have any difficulty in suggesting the data structure in a given scenario. They have been able to convincingly justify their choices. A few answers had supporting diagrams complementing students' understanding.

Some of candidates had difficulty in making the right choice of the data structure required while answering a question. They failed to identify any data structure at all while the question was very specific about naming the right data structure. Candidates were expected to endorse their choices by explaining why they chose a particular data structure.

Question 20

Some candidates had difficulty in understanding that **inheritance** is the capability of a class to use the properties and methods of another class while adding its own functionality; **and**

polymorphism is the capability of an action or **method** to do different things based on the object that it is acting upon. These are the basic principles of object oriented programming. If candidates have learnt their basics well, these questions should not have troubled them.

Question 21

Almost all candidates constructed correctly the truth table in Part (a). Many candidates used K-diagrams to minimize the Boolean expression. Answer to Part c depended on the simplified expression given in answer to Part b. A few candidates were not able to draw a circuit for the expression.

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

The more examination practice the candidates get the better. This does not imply that they have tests continuously through the course, but examples of previous questions could be integrated into the teaching and learning process. This will allow for reinforcement of subject matter and make them familiar with how the questions are structured and asked in the examination. It also helps to have a copy of the syllabus distributed to each candidate at the beginning of the course, then they are able to ask about topics that they find difficult.

Particular attention should be paid to the action verbs used in the questions; Avoid generalized answers and specifically answer the question asked.

Higher level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 17	18 - 34	35 - 41	42 - 51	52 - 61	62 - 71	72 - 100

The areas of the program that proved difficult for candidates

All questions proved to be accessible to most students and there was little indication to show that lack of time was a major factor in this exam. The top marks were in the low 90s.

The recursion question in Question 2 (part d) proved the most difficult as was anticipated and challenged the better students. File-handling was also an area which proved difficult (with some schools more than others).

The levels of knowledge, understanding and skill demonstrated

Most students showed a reasonable understanding of all the topics tested. Manipulation of arrays and the use of OOP techniques have improved considerably over the last few years. The students showed a good general knowledge of most of the Case Study topics.

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

Question 1

The question was based around an array of objects which were subjected to two standard searches: sequential and binary.

The success of the students in dealing with the two algorithms underlines the improvement that schools have made in preparing their students for these types of questions. The sequential search gained full marks for many candidates. Students now seem comfortable in dealing with arrays. The fact that the first element has an index of 0 no longer causes problems when determining loop conditions. Although no marks were deducted for lack of efficiency, most terminated the iteration as soon as the target was found.

The binary search proved slightly more difficult. Minor errors included setting incorrectly the new top and bottom values.

It is noticeable how the majority of students are now comfortable with the manipulation of objects, something which has improved considerably as teachers have incorporated the main features of OOP into their courses.

Question 2

With Question 1 dealing with SL theory, this question focused on the HL course, in particular binary search trees.

Parts (a) - (c) provided an opportunity for most students to gain marks, showing that the fundamentals of tree construction and traversal were understood.

Dealing with traversal at an algorithmic level proved more difficult, as was shown with the node counting question in part (d).

The question required the students to traverse the tree recursively either returning 1 to the previous level each time the new root was '*not equal to null'* or incrementing a counter. Recursion is a difficult topic for most students, but even so there were several students who gained full marks, Credit was given for those students who understood the basic elements of a recursive algorithm: the need for two cases, one being a terminating condition (*if root = null*), and one which calls the method with a changed parameter *(new value of root*).

Part (e) required the students to add a node in the correct place in the tree.

Most made a reasonable attempt at this, but many omitted to deal with the empty tree case. Some were confused as to the difference between a reference to a node and the node's data members, and consequently made impossible comparisons. Others failed to use a temporary node to traverse the tree. A few students answered this recursively which was not anticipated, but the resulting shortened code showed how elegant recursive solutions can be.

Question 3

This file-handling question proved the most difficult question on the paper indicating, perhaps, that some schools are unable to devote adequate time to this topic towards the end of the course.

Once the hash table has been formed, a record ID number can be used to directly access the record on disk. The position of the record is determined by applying the same algorithm used in part (d).

Most knew what a key field and how to find the hash key for the hashing algorithm. The two main problems were in determining the data structures that could be used - the array in part (b) and especially the alternative structures in (f), where the expected answers should have dealt with indexed files - and in understanding why a hash table was used at all. Many students expected the whole (large) file was to be found in the main memory location indicted by the hash algorithm, and did not realize that the hash table would point to the record's location on disk.

Although students could refer to different ways of resolving collisions (free space, chaining, probing), not all clearly indicated that the extra steps require a sequential process until the correct item is identified.

Question 4

The Case Study question was answered well by all students, and was indeed the salvation for some. Provision of equal opportunities for all members of society is an important topic which was clearly acknowledged by the students through their answers, some of which were quite innovative. Students should always be aware of the number of marks available when answering a question. The part which gave most problems was part (g) (social and economic advancement), which was only answered well by those who had presumably spent time discussing the aspects of the Case Study in detail.

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

- Deal with file-handling in sufficient detail.
- Algorithms should be covered for all parts of the syllabus.
- If an algorithm has a return value in its signature then a value should be returned and not output.

- Similarly, parameters should be used to pass data to a method as opposed to entering them from the keyboard.
- Particular attention should be paid to the action verbs used in the questions; Avoid generalized answers and specifically answer the question asked.
- Be familiar with all scenarios and terminology provided in the Case Study.

Standard level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 11	12 - 23	24 - 30	31 - 37	38 - 43	44 - 50	51 - 70

The areas of the program that proved difficult for candidates

Many candidates performed reasonably well. Schools have to reiterate the fact that precise and specific answers are required of their candidates.

The levels of knowledge, understanding and skill demonstrated

There is a wide knowledge base of candidates appearing for this examination. The syllabus seems to be covered by most schools, the performance of many candidates hovered above the average. There were a few students who were either outstanding or very poor in their performance.

Many candidates write answers that are very brief and tend to neglect the amount of marks allocated to each question.

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

SECTION A

All of the questions in this section seem to be accessible to all the students and the majority showed a good coverage of the course.

Question 1

Many candidates simply listed characteristics of the analysis and design stages but neglected to describe differences between the two stages. Note that the action verb requires an explanation, not simply a list.

Question 2

Many candidates limited their answers to listing advantages rather than explaining them. Others considered only the advantages of modular design, and thus failed to note advantages in the construction, testing, and maintenance of computer programs.

Question 3

Many candidates failed to achieve full marks as result of incomplete, non-specific answers. The question required a description for each sub-part and that description required the identification of a specific computer system/application and its effect on a well-identified group of people.

Question 4

This question was generally well-answered

Question 5

This question was generally well-answered.

Question 6

Some candidates simply proposed the use of two arrays for part (b) without identifying any mechanism for keep the two elements associated with a particular person aligned. A surprising number of candidates proposed using a two-dimensional array to store both ints and booleans.

Question 7

This question was generally well-answered.

Question 8

This question was generally well-answered.

Question 9

Many candidates identify the end-user of the completed program rather than the programmer as the person who writes a user-defined method.

Question 10

Parts (a) and (b) were generally well-answered. Some candidates incorrectly predicted changes to the array if it was already sorted in part (c). Many candidates neglected to identify a need in part (d) for the size of the array to be determined by the method at runtime, or to suggest a means by which this could be accomplished.

Question 11

Very few candidates correctly answered this question. The use of a transaction file to update a master file within a batch process appeared to be unfamiliar to many candidates. In part (a) many candidates failed to make any specific reference to the data that would be in a payroll system and instead attempted to write completely general descriptions of master and transaction files. In part (c), a surprising number of candidates identified accidental data-entry errors rather than deliberate errors made by a conscious effort.

Question 12

In part (a) many candidates asserted that smaller files can be transmitted faster without differentiating between transmission speed and transmission time: The smaller files take less time to transfer because there is less data to transfer, not because the data transmission speed is different. In part (b) only a few candidates where able to explain the role of protocols in a network.

Question 13

This question focused on the use of a microprocessor embedded within a physical device. In part (b) however, many students identified a system that was simply too large or too vaguely described to be considered a device. In part (c) many candidates failed to recognize that when a microprocessor is embedded in a device its inputs and outputs are connected to the devices hardware and instead described user-related inputs and outputs.

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

The more examination practice the candidates get the better. This does not imply that they have tests continuously through the course, but examples of previous questions could be integrated into the teaching and learning process. This will allow for reinforcement of subject matter and make them familiar with how the questions are structured and asked in the examination. It also helps to have a copy of the syllabus distributed to each candidate at the beginning of the course, then they are able to ask about topics that they find difficult.

Particular attention should be paid to the action verbs used in the questions; Avoid generalized answers and specifically answer the question asked.

Standard level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 11	12 - 22	23 - 30	31 - 37	38 - 43	44 - 50	51 - 70

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

Question 1

This question was reasonably answered.

A reasonable number of students handled the algorithms well and showed a sound understanding of passing values, handling arrays, performing iterative algorithms to accumulate totals, defining methods/functions, returning values and handling data types. However, a lot of students demonstrated a complete lack of facility in this obviously important area of algorithm and computer program development.

Question 2

This question was reasonably answered – students on the whole knew what a sequential file was.

This question worked well with a number of students being able to outline the basic logic needed. However, it was not clear that students appreciated that the file had to be re-written,.

For question (d) good students seemed to know what to do, but a number of students did not seem to have had much exposure to defining basic records.

In a number of cases question (e) was not answered well and could indicate a gap in the teaching in some schools. However, it seems that the dossier at SL is still easiest done by addressing a classic file handling problem which requires a record that is either managed one record at a time with sequential access/organisation and/or read into a list structure and sorted, searched and the file updated. Therefore, it is still surprising that a number of students are not able to address this type of problem.

Many students handled question (f) reasonably well.

The algorithm was familiar to a number of students, and they were very pleased to expand upon this knowledge by providing very nice and lengthy descriptions of the binary search as an answer to (h).

Alas, many students did not read the words ‘how used’ and hence failed to answer the question.

Question 3

Many students gained the majority of their marks on this question.

Whilst the questions might be viewed as straight forward, it was clear that students understood the importance of the issue addressed in the case study.

Students on the whole demonstrated a good capacity to construct meaningful written answers.

The final question caused some problems in terms of marking as many students did not name the particular type of technical documentation but were able to describe a strategy.

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

- Algorithms should be covered for all parts of the syllabus.
- Particular attention should be paid to the action verbs used in the questions; Avoid generalized answers and specifically answer the question asked.
- Be familiar with all scenarios and terminology provided in the Case Study.