International Baccalaureate®
Baccalauréat International
Bachillerato Internacional

COMPUTER SCIENCE

Overall grade boundaries

Higher level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 14	15 - 28	29 - 41	42 - 53	54 - 65	66 - 77	78 - 100

Standard level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 15	16 - 32	33 - 43	44 - 54	55 - 65	66 - 76	77 - 100

At HL, both the number and quality of candidates were similar to November 2006. Only 21 candidates sat the exam, but the quality of these candidates was high, with a larger percentage of candidates achieving grades 6 and 7 than would be normally expected. Weaker schools seemed to have dropped out of HL by 2006, the main bulk of schools now being from Australia (traditionally strong in this subject). Added to these are a few (newer) schools from S. E. Asia.

The number of students taking the SL exam was 51, a number similar to November 2006.

Higher and standard level internal assessment

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

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

Most of the dossiers seen in this session were well-structured and organised and the selection of topics was generally appropriate. Any problems usually seemed to stem from:

- Poor choice of a “closed problem” – typically gaming related
- Not having a real user or not talking to the user about the information problem itself
- Coding before design

The majority of students now do seem to be making use of a real user, and making some effort to ‘collect information’ often in the form of interviews. The interviews are not always backed up with documentary evidence of data collection.

The most surprising aspect in the session was the number of students that failed to achieve a mastery factor of 1, which would seem to be a fairly easy task, at least at Standard Level. Quite a few students at this level are failing to use any data structures (e.g. arrays of objects) to hold their data for processing in memory, instead calling up records one at a time from their file(s).

Also surprising was the number of programs that failed to work (by the admission of the teachers). Some of these dossiers that fail to achieve full mastery marks and/or didn’t work included quite detailed graphical interfaces.

A pleasant change is the addition by many schools of a separate document detailing the teacher’s reasons for awarding their mark for each criterion. This helps greatly in moderating and also allows the moderator to give good quality and effective feedback.

Candidate performance against each criterion

The analysis in A1 was, at times, unconvincing with users appearing to be mythical or at least remote from their own problems. Students who have a sound grasp of the user’s information requirements as opposed to looking through the filter of a programming solution tend to handle this section better.

Objectives should be specific and testable and not vague.

The **prototype** requires an initial modular design – this is a good way for candidates to begin to clarify both the information requirements and the potential solution and should not be neglected or hurried. It can be a very valuable exercise when done well.

For section B, the **algorithms** and **modules** are expected to be written before any code is produced and many dossiers show evidence of simply cutting and pasting code and/or class information from the program listing. Standard GUI objects are not generally recognised as data structures in the sense intended in the criterion B1. Students who develop record-type classes using only String data are in danger of not being awarded mastery – some variation in the use of types can be expected (otherwise why use a record at all).

For the **data structures section** candidates should use sample data from the chosen problem domain rather than supply abstract diagrams of arrays, lists and files – especially where candidates all decide to use exactly the same list structure and exactly the same file-handling techniques.

Candidates’ presentation of the **code listing** varied from very good to extremely weak and confused. The best examples used syntax highlighting, carefully separated Class definitions and clear comments on mastery aspects. Good candidates also carefully distinguish their own code from that supplied by teachers or automatically generated by an IDE.

In the **sample runs** section, D1, a common problem was either a lack of sample runs or an excessive degree of testing of invalid data. Either way it was difficult for the moderator to confirm mastery and other aspects awarded by the teacher. The subject guide requires at least one complete run of the program and this should show that updates such as additions and deletions work was expected.

A common consequence is that the **effectiveness of the solution** (C4) is also called into question and claims in the evaluation section about what actually worked are also harder to evaluate.

Sample screenshots are required for the **user documentation** section.

Mastery Aspects

There continues to be much misunderstanding of mastery aspects, particularly at Higher Level. Most teachers do not seem to have read the May 2006 subject report which deals extensively with this issue. The major problems seen in this session relate to:

Use of the **RandomAccessFile Class** was not always convincing. Candidates must not simply stream data from an array or list into the file, and should not seek to file.length() and then append a record as the only way of adding. If the records are unordered, then re-use of space taken deleted records should also be implemented.

Parsing a text file or other data stream – it is not sufficient to use the methods of the wrapper classes or the RandomAccessFile Class (eg Integer.parseInt() etc). The candidate has to read some text and then take an action or set a value based on the text that has been read.

Inheritance is considered trivial when simply extending a JFrame or similar GUI object. The candidate could develop special GUI or Exception objects which inherit from existing ones as long as they describe and justify this approach in B1/B2.

Polymorphism is not achieved by overwriting the Object toString() method.

Recursion should not be used where an iterative approach would serve equally well. It is certainly not appropriate for candidates to devise a recursive input validation return with no visible terminating condition.

When developing an **ADT** the need for each method should be discussed in terms of the solution requirements. Also, each method supplied should actually be used within the program. Needless to say, it is not acceptable merely to present code from a book or website.

Recommendations for the teaching of future candidates

While teachers and students alike are very fond of models showing the “right way” to approach a dossier it might be wise to avoid an overly prescriptive approach - for example, copying elements from the Teacher Support Material is not a good idea. The new guide is designed to offer candidates flexibility in their approaches to analysis and design.

Problems should not be overly simple or overly complex. About 4-6 specific, measurable, achievable, relevant and time-constrained (SMART) targets will be enough for section A2. These should, of course, be referred back to most stages of the design and development process and form a critical feature of a good evaluation.

Quite a few candidates are apparently running out of time to complete the solution. Students should be encouraged to concentrate on basic aspects first and add any special features later. Getting a working program shell using "stubs" and then expanding each of these is an excellent way to achieve this.

Further comments

Quite a few clerical errors were noticed and teachers should be careful in:

- the addition of marks on the 5/IACS
- not exceeding the maximum available mark for each criterion
- ensuring correspondence between mastery aspects on 5/PDCS and factors entered on 5/IACS
- Making sure the page numbers for mastery factors are correct.

Higher level external assessment

Both exams produced a good spread of marks, thereby differentiating between the candidates. However, the exams were perhaps a little easier than last year's, in particular the students generally scored high marks on the Case Study question in Paper 2 (40% of the paper.

Higher level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 14	15 - 28	29 - 48	49 - 56	57 - 64	65 - 72	73 - 100

General comments

There was a small number of candidates, but of generally high quality, with few weak students. The paper was straightforward and certainly accessible to those who had a good basic knowledge of the course. The paper produced a reasonable spread of marks, with the majority achieving grades 5/6/7.

The areas of the programme and examination that appeared difficult for the candidates

- double buffering
- floating-point

The areas of the programme and examination in which candidates appeared well prepared

- prototyping
- instruction cycle
- interrupts – definition (not functioning)
- types of file processing

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

Section A

Question 1 prototyping

Generally answered well. All students knew that a prototype was a partly functioning model.

Question 2 Linker

Most had the idea of allowing different modules to work together. Not all knew how.

Question 3 documentation

Some confused this with systems software. Some, interestingly, included user documentation with systems documentation.

Question 4 instruction cycle

Students know this well.

Question 5 double buffering

Most guessed that there were 2 buffers and that this was a good thing, but the majority didn't understand the concept.

Question 6 interrupts

Almost all were able to give a good definition, but the explanations were not all clear enough.

Question 7 batch / real time

Students generally have problems with compare questions, describing the issues individually without performing any comparison, as was the case here. They would be better using a table format, where they could outline side-by-side the differences for particular characteristics.

Question 8 types of file processing

Most answered correctly.

Question 9 recursion

A simple enough recursive algorithm, leading to a high number of correct answers.

Question 10 bar codes

Some answers were too general and dealt with any method of data collection, e.g. allows items to be counted, but most answered well.

Section B

Question 13 searching / sorting

Very easy question with many candidates gaining full marks. Some had a problem with the final part, and although many suggested a BST, not all followed this up with a specific reason,

Question 14 WAN

(b) **advantages/disadvantages** - Candidates should avoid general answers such as “easier to buy goods” or “more convenient”, and instead focus on answers that are specific to the scenario given.

(c) **packet-switching** – Most understood this, even if they didn’t manage to get the full 4 marks.

(d) **serial/parallel** - Another comparison question – either deal with both methods in the same sentence or use a table.

Question 15 Transaction/Master Files

(a) Data structure answered well.

(b) Better than normal, but some candidates continue to be unsure about system flowchart symbols.

(c) Question requests the differences between transaction and master files, **not** just a description of each. Candidates found difficulty in making the differences clear.

Question 16 Linked Lists

Not really a linked list question, with only the last part dealing with dynamic structures. The main part dealt with a static structure and was quite easy.

Question 17 Databases

(a) Data Input: most rightly chose an automatic method, although some believed entering hundreds of forms manually was ok.

(b) Verification – no problem.

Question 18 Number Representation

This was a good HL question. Most students knew how to convert from binary to hexadecimal but few knew why/where hex. is used, with quite a few believing that it allowed a greater range of numbers to be stored in memory.

(c) The floating point question gave problems as always – it allowed the 6/7 students an opportunity to shine.

Recommendations and guidance for the teaching of future candidates

Keep teaching the basics well (as was shown by the majority of the candidates). Certain HL concepts such as double buffering, floating point and interrupts need special treatment.

Students generally have problems with compare questions (14(d) – parallel/serial), describing the (2) issues individually without performing any comparison (and thus losing marks), as was the case in all the papers. They would be better either using a table format, where they could outline side-by-side the differences for particular characteristics, or compare the same characteristic for both methods in the same sentence.

Similarly 3(c) requested the differences between transaction and master files, **not** just a description of each. Candidates found difficulty in making the differences clear.

As always, action verbs need care. In 13(c) many students correctly suggested the use of binary trees as a way of avoiding sorts, but failed to explain **why**, as the questions asked. Teachers should as a matter of course use the IB action verbs throughout the course to allow students to familiarise themselves with the appropriate type of response for each verb.

When a scenario is given (14(a) - WAN), students should take care to avoid general answers such as “easier to buy goods” or “more convenient”, and instead focus on answers that are specific to the scenario given.

Higher level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 17	18 - 35	36 - 43	44 - 54	55 - 66	67 - 77	78 - 100

General comments

As mentioned above, the overall standard of the candidates was high. In Paper 1 they showed a good overall knowledge of the course. In P2 they performed well on the (two) algorithm questions.

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

Question 1 – Board Game

The high standard in dealing with algorithms was shown in 1(d), with a surprising high number producing correct methods for setting the restricted array elements to -1. It’s always interesting to see the diverse ways that students (correctly solve problems). In this case some realized that the sum of the row and column elements was always even, and so used modulo arithmetic. Others looped though even rows first and then the odd rows using increments of 2.

The other two algorithms to return the empty squares and to test for legitimate moves were straightforward.

Not all came up with a suitable method for storing the status of the board with less memory (f). Some came up with quite elaborate methods that would have been difficult if not impossible to program.

Question 2 – Doubly-Linked List

Parts (b) and (c) were a good test for HL students, as re-assigning the links between nodes when a node is deleted needs a thorough understanding of how pointers work. The teaching of this topic needs a lot of careful thought, but it was clear that most of schools involved had achieved this.

Most, perhaps, not surprisingly came up with the wrong answer for the BigO efficiency of the delete process (d), but most were able to give an application of a queue. Those who got this wrong described a list (static) situation, instead of an application where elements can join as well as leave.

Question 3 – Files

Most understood the concept of a balanced tree, as shown by the ’efficiency’ answers in (c).

There were interesting alternative answers to (d), with most opting for a second BST, with others sorting the original file by ID number and keeping only the original tree.

Part (e) provided many detailed descriptions of partially indexed files.

Question 4 – Case Study

This question proved straightforward for those who had studied the Case Study material. Students should be carefully about making reference to certain pages in the Study, ensuring that the complete answer is on their exam paper.

Recommendations and guidance for the teaching of future candidates

Continue to encourage students to come up with and share different methods of solving problems (as shown above in the choice of algorithms and organization of files).

Ensure that enough time is given to studying and discussing the Case Study material. There were a few examples in this exam of students achieving highly on the HL material, but faring less well on the less challenging Case Study questions.

Equally, students should not underestimate the time required to complete this question (which is the last on the paper), as it is worth double the marks of any of the other questions.

Standard level external assessment

Both papers produced a good spread of marks, thereby differentiating between the candidates.

Standard level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 12	13 - 25	26 - 32	33 - 39	40 - 45	46 - 52	53 - 70

Some students were weak on the general theory with disappointing marks in Section A. This suggests that some schools have not got the balance right between the time devoted to theory and the time devoted to programming. This includes schools that have been running the Computer Science program for some years. It was clear that certain topics had not been studied at all, as was shown by manner in which these questions were answered.

On the other hand, other schools clearly have developed a competent, balanced course, and consequently the students from these schools perform well each year.

There was a wide spread of marks ranging from minimal scores to the mid 60´s (out of a total of 70). All questions appeared to be accessible and there did not seem to be any time problems.

Section A

Question 1 – Levels of Security

The phrase ‘level of security’ was not understood by several students, who instead answered generally about security issues. The most popular answers for those who understood what was being asked referred to ‘the number of users’ or ‘the type of network (if any).

Question 2 – Case Tools

There were few correct descriptions of CASE tools.

Question 3 – Online Processing

Most were able to give examples (b) without correctly identifying the characteristics (a).

Question 4 – Syntax Errors

This was answered reasonably well, although many referred to ‘problems with the code’ which is too vague an answer. A more detailed answer should refer to ‘errors in the rules for writing code’ together with an example.

Question 5 / 6 – Feasibility Report / Cache Memory / Virtual Memory

These two questions highlight weakness in knowledge of the basic theory, as many candidates seemingly did not recognise these terms.

Question 7 – Direct Access

Most knew that direct access was faster (a) but from the range of unusual answers (b), not all knew why.

Question 8 – Modularity

This was better answered, with most students able to give at least one good answer.

Question 9 / 10 – Interpreter / Data Compression

Again, seemingly easy questions asking about basic theory, but not many students gained full marks.

Students must learn to avoid using the terms in the question as part of their answer. For example, ‘the data is compressed so that it takes up less space’ would only gain 1 mark.

Section B

Question 11 – LAN

This was a straightforward question and answered well by most students. Students must carefully read each question though – part (a) referred to **both** hardware and software, and students had to refer to both to gain full marks.

Question 12 – Security / Data Integrity

Part (a) (Security Measures) was answered well, but not all were able to state two methods in (b) for ensuring data integrity. Students continue to confuse *security* with *integrity*.

Candidates need to think about specific marking points in ‘Discuss’ questions. For example, ‘*people may break into the system and steal data*’ is not as specific an answer as ‘*unauthorised users may break into the system and steal data which could lead to identity theft’*.

Question 13 – Prototyping

The question was generally answered well, with candidates familiar with the idea of prototyping through the Case Study.

The way in which students interpret the action verbs are crucially important. All questions will use one of the action verbs described in the subject guide, and will expect an appropriate response. For example, in part (a) (i) students were asked to **compare** *questionnaires* and *observations*. Describing the two methods/terms completely independently of each other is unlikely to gain full marks in this type of question. Students should either describe the same characteristic for each method (in this case) in the same sentence where possible, or make use of a table where a description of same characteristic for each method is put next to each other,

For example.

Questionnaires are a quick and cheap method of collecting data, whilst observation can be time consuming and therefore expensive.

or,

	questionnaires	**observation**
cost	Cheap to carry out	Time consuming, therefore expensive

Question 14 – Algorithm

There was a wide range of marks awarded, although the performance of students on this type of question does seem to have improved over the last 2 or 3 years (but sometimes at the expense of the theory).

As always, students need to look at the number of marks awarded (particularly with respect to parts (b) and (c) in this question). There were several students who were clearly able to answer the questions but who failed to provide 3 distinct points for each part.

Standard level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 - 12	13 - 25	26 - 30	31 - 36	37 - 42	43 - 48	49 - 70

As with Paper 1 there was a wide spread of marks, but this time there were fewer marks in the extreme ranges (very high or very low). All questions seemed accessible, but there were a small number of students who didn't finish the paper.

As always the algorithm questions produced the complete range of marks, with many very good (and varied) solutions. The students showed that there were several ways to solve the same problem, and this varied approach should be encouraged in schools.

The Case Study proved particularly accessible to most students and accounted for the majority of marks for the weaker students.

Question 1 – Sort Routine

(a) The trace table was generally well done. The students seemed to be well-practiced at this type of exercise.

(b) Efficiency

Most knew that this particular sort routine was a bad choice. The better students were able to gain the second mark either by saying why or by referring to more efficient routines.

(c) Early exit for the routine

There were many full marks here, and most students gained some credit. If a mistake was made, it tended to be the failure to reset the Boolean variable at the start of each pass.

(d) Insertion routine

This is where the good programmers came to the fore with several, equally valid solutions.

Question 2 – Intelligent Bus Stops

(a) Most students stated two items that would be displayed.

(b) A surprising number did not realize what was being asked, and went on to describe at length the processing that was going on. This is a standard *state* question that only gains 1 mark and generally requires only 1 or 2 words in the answer (in this case – *real time*).

(c) Almost all read the correct answer from the able, but only the good programmers seem to recognise why the value **-1** was in certain places. Apart from being placed in the ‘impossible’ combinations, the value is not one that could be an actual journey time, which it could be used as a type of sentinel in a program (as in part (e)).

(d) There were 3 marks available here for explaining why the system should be tested thoroughly, so an actual consequence of the result of insufficient testing should have been included in the answer (for example, ‘*times are displayed incorrectly so passengers stop using the bus system leading to lack of revenue for the bus company*’).

(e) The findTimes method.

Again, there were a variety of correct ways of writing the method, and again this was an opportunity for the good programmers to gain marks.

(f) There were a lot of good answers .

(g) To gain 3 marks for an expansion to the system required a detailed answer, describing the expansion, stating the benefit and explaining why this would be a benefit.

Question 3 – Case Study

This question proved straightforward for those who had studied the Case Study material. Students should be carefully about making reference to certain pages in the Study, ensuring that the complete answer is on their exam paper.

Recommendations and guidance for the teaching of future candidates

Some schools clearly have well thought out courses and continue to enter candidates who are successful in these exams. Other schools, however, need to look carefully at their courses to ensure that both theory and programming are each given sufficient attention. When dealing with the programming, it is essential that problem solving takes precedence over other aspects that are not tested in the exams (such as producing GUIs).

Action verbs need to be carefully discussed and used throughout the course (with the correct type of response expected). Students should practice matching their answers to the number of marks awarded for each question.

Ensure that enough time is given to studying and discussing the Case Study material. There were a few examples in this exam of students achieving highly on the SL material, but faring less well on the less challenging Case Study questions. Equally, students should not underestimate the time required to complete this question (which is the last on the paper), as it is worth more than the other questions (30 out of the 70 marks).