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 - 39	40 - 51	52 - 64	65 - 76	77 - 100

Standard level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 15	16 - 31	32 - 43	44 - 54	55 - 65	66 - 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

A broad range of work was submitted and, in general, the level of work was acceptable. Also, in some regions there was a noticeable improvement in overall quality. Nearly every dossier was well-structured with useable tables of contents and the correct sections.

The choice of projects was very varied at this level and it is encouraging to see candidates attempting non-file-based examples. Many of these were very successful.

Some types of problem were far too complex for students at this level to attempt and, as ever, games programs were usually a poor choice for all but the most able.

There was greater use of event-driven and graphical programs demonstrating a sound understanding of OO design methods, which is encouraging. There were times, however, when the overuse of IDE generated code led to a cluttered interface with a large number of popup windows with minimal function obscuring both code and hard copy printout.

Candidate performance against each criterion

Criterion A1

Generally better this year with fewer candidates describing their solution instead of the initial problem. There were quite a lot of obviously imaginary end users with understandably limited advice to offer the candidate.

Candidates who had a genuine problem to analyse and an intended user with whom to discuss the solution did much better.

Criterion A2

This criterion continues to cause problems for some candidates. Candidates often failed to relate their goals to the analysis made in A1. To design an effective solution to a problem requires clear, well-written, specific, provable or testable goals. For a higher level dossier there should not be more than 6–8 non-trivial examples of such statements.

"I will design a system to track the products through the warehouse" is vague and not easily demonstrated.

"When products enter or leave the warehouse, the product ID will be used to locate and update the stock levels in the product file" is a demonstrable objective of a system.

"This will allow the user's objective of keeping accurate records of materials on hand in the warehouse to be satisfied" would tie in nicely with the analysis section.

Criterion A3

A significant minority of dossiers presented screen shots, clearly coming from the functioning program, as prototypes. These candidates find it almost impossible to score well in this section.

Good prototypes were completed in both presentation and drawing packages. Weaker ones were hand-drawn and unclear.

It is possible to design and then code a prototype, using Java or any other convenient language, and this has the advantages of providing some interaction and giving further insights into the eventual solution. However, the code is best relegated to an appendix if the candidate feels the need to include it.

A prototype design should be a simple module diagram or list very briefly indicating the main functions/screens to be designed. Feedback from the user should be more realistic than, say, "Wow, you really know how to produce a great design". Experienced designers hardly ever get everything right first time.

Criterion B1

Many candidates used data structures which were appropriate. The largest problem is probably using generalized illustrations of lists, stacks, queues and trees which do not include data or operations on the data structures that are relevant to the problem being described.

Diagrams of a general nature are not required.

Candidates at higher level should be able to discuss a range of data structures giving detail such as sample data and sketches of those they actually intend to employ in the solution.

Some candidates prefer to present JavaDoc or even code listings for this section and this is inappropriate.

Criterion B2

Some candidates continue to present Java, copied or slightly edited from their completed code, as examples of their algorithm design. This practice teaches the candidate nothing but uses up vast amounts of their precious time.

Algorithms should be presented in sufficient detail so they can be coded from the description.

There was confusion as to the purpose of pre and post conditions and some teachers are advised to study this topic in a little more depth. A detailed understanding is not required but the candidates should at least know that the conditions are “contracts” made by the programmer and not the user.

Criterion B3

Connections between modules were often apparent, but connections to algorithms data structures less so. Often this can be achieved by a sensible system of naming and by having an overall (structured) diagram accompanied by brief descriptions of each module—what it does and what it does it to. Obviously it is not necessary to repeat the algorithms and data structure details again so for a higher level dossier this might typically 3–4 pages.

It is worth pointing out that candidates do not need to revisit sections A and B because they changed their minds at coding time; as long as sections A and B are “complete, logical and useable” candidates can deviate from it in sections C and D.

It is true, however, that this deviation may have an impact on subsequent criteria that depend upon the goals of section A2 (C2, C4 and D1). This impact will usually be small and letting those three (approximately) achievement levels go, will save candidates a lot of time in unnecessary back-tracking to the analysis and design stage.

Criterion C1

Code layouts were generally good with a broad range presented. Candidates should be reminded to pay attention to the formatting of code pasted into word-processed documents. Using landscape orientation for the code is recommended.

Mastery aspects should be commented on in the code as page numbers sometimes appear to change between indexing mastery and printing the final dossier. It is extremely frustrating for the moderators to have to search through long listings to find evidence of mastery. This is especially frustrating at higher level where the algorithms and data structures can be very complex.

The page numbers must refer to the actual code listing and not to other sections, for example, the mastery index. It is always sad when candidates fail to receive credit for something they have achieved because they, or the teacher, did not take the necessary care over administrative details.

Criteria C2 and C3

These criteria were generally well documented. Many candidates provided a systematic account of usability and error-handling and the best examples were well-structured in either making references to code lines and hard copy printouts (which were themselves clearly labelled) or cutting and pasting from code and hard copy into this section. Either way works well.

Criterion C4

Some teachers continue to award high marks for an effective solution that has not met most of the criteria outlined in A2 or has not demonstrated capability via hard copy.

Criterion D1

The best sections included tables showing what was being tested in the various screenshots and demonstrated a systematic approach by the candidate. Once again, it was common to provide more abnormal/invalid data testing than actual shots which demonstrate the solution works as claimed. Candidates should provide at least one run showing that features such as adding, editing, sorting, saving and deleting items from lists, trees, files, etc actually work.

Criterion D2

Most candidates discussed possible improvements and effectiveness. More successful candidates also presented discussions of efficiency.

Criterion D3

Some candidates failed to provide illustrations or installation instructions but this section was well done on the whole.

Recommendations for the teaching of future candidates

Sections A and B continue to provide candidates with opportunities to demonstrate their problem-solving and analytical skills. There is no doubt that these show signs of improvement at this level, at least where schools are clearly encouraging candidates to complete these sections first. Probably there is a 2/3 grade gap between those schools and schools where candidates take a completed solution and then produce the analysis and design. Admittedly, there is no quantitative evidence for this statement.

The selection of a task should be done such that it provides enough challenge to keep candidates' interest and yet be achievable. Overly complex solutions must be simplified at the earliest possible opportunity and teachers should not allow candidates to get themselves in too deep.

Teachers need to constantly monitor candidates to ensure they have not got side-tracked by minor issues and are staying on schedule.

Both the problem and the proposed solution should be developed with a clear idea of how mastery of aspects is to be achieved.

Teachers and students should understand that mastery is not simply the use of a required mastery aspect in code. A standard sort is not evidence of mastery of sorting, loops, selection or anything else without documentation demonstrating the candidate's understanding of the algorithm and hard copy evidence that it actually works.

Problems continue to arise from the granting of mastery for:

- polymorphism when simply using multiple constructors or overriding built-in methods (eg toString())
- parsing a data stream when simply using Integer.parseInt() for numeric conversion
- inheritance from the Frame or JFrame Class
- random access files when simply using a text file.

Please refer to the May 2006 Subject Report for clarification of mastery issues before advising candidates on the dossier.

In some cases GUI programming, often using code-generating IDEs, appears to be getting in the way of producing good dossiers. Many of these dossiers are full of event-driven graphical objects with little purpose and which make little contribution to demonstrating mastery but serve to clutter the code and obscure the work the candidate has actually produced—especially when presented in the hard copy section. The password dialogue is a case in point: no practical value is served by these in the majority of student dossiers. Text-based solutions which suddenly pop up error message windows are another pointless feature increasingly encountered in dossiers, unfortunately.

Further comments

Teachers and candidates should take care to check that page numbers given on form 5/PDCS actually correspond to page numbers in the code listing. Where teachers do not tick these boxes and yet the code shows apparent mastery of aspects, the moderator will generally not award the aspect, assuming that the teacher has his/her own reasons for not crediting the aspect to the candidate (eg the code in question was developed in class and never modified or understood by the candidate). Of course, it would help moderators if teachers remembered to make notes to this effect on the work or on the 5/IACS form.

A really helpful dossier would have the "mastery index" make reference to the code listing and the page where the use of the aspect is documented (in section B).

Overall, the standard of work was higher this session than last and quite a few schools are turning in high-scoring dossiers of around 80–100 well-written pages. Some schools continue to submit dossiers filled with excessive material (code, hard copy or JavaDoc are the main culprits) which add to the burden of students without contributing marks. Some of these dossiers exceed 250 pages which is a sad waste of a student's time in many cases.

The examiner would like to make a special plea to those teachers, coming in to the subject without a solid computer science background, to consider acquiring the necessary skills as a matter of urgency. Too many teachers do not have sufficient background to understand the mastery factors themselves and appear content to rest on the candidates' own judgement.

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

A broad range of work was submitted and, in general, the level of work was acceptable. Also, in some regions there was a noticeable improvement in overall quality. Nearly every dossier was well-structured with useable tables of contents and the correct section.

The majority of the samples were basic databases. Ideally, candidates who set out to write a database type solution should be given some kind of background in the design of suitable files. The choice of suitable key fields and appropriate data types is to be encouraged. However, candidates at this level should not attempt overly complex designs involving multiple files.

Other types of problem met with varying degrees of success and, as ever, games programs were usually a poor choice.

There were a few candidates that used an Access database and therefore failed to show mastery of objects as data records, arrays, searching and/or sorting. The teacher wrongly assumed they were showing mastery in those aspects, but they were only using the facilities provided by Access and SQL instructions instead of producing their solutions. Using Access and SQL is permitted for mastery of file i/o however, as long as additions, edits and deletions are made and documented.

When using standard file i/o the java.io.* library routines should not additionally be claimed for mastery of additional libraries.

A sentinel value must be explicitly declared and used to terminate a process appropriately, not for example, as a simple counter in a loop.

The most common method seemed to be a procedural approach, often coupled with console programming. While this may be a good choice for the weaker candidate one might imagine that more able candidates would get more out of attempting an event-driven and object-oriented design. Either method, however, is appropriate for the present syllabus.

Candidate performance against each criterion

Evidence from form 5/PDCS seems to indicate that a significant number of teachers are requiring stages A and B to be completed well in advance of the actual coding. Nonetheless, candidate performance overall remains weak in stages A and B.

Criterion A1

This was generally done better this year, although some candidates still slipped into a description of their solution without mentioning what the initial problem was. Also, there were quite a lot of imaginary end users. Candidates who had a genuine problem to analyse and an intended user with whom to discuss the solution did much better.

There were some candidates who insisted on describing the solution rather than the problem and this usually prevented them from seeing the actual problem clearly. This approach also makes it difficult for them to establish clearly enough areas where they can make a logical connection to the A2 criteria for success.

Criterion A2

This criterion continues to be a problem. One big sticking point is that candidates rarely explicitly relate their goals to the analysis proposed in A1—as noted above, this is often because the solution and not the problem has been described in A1.

Also the goals offered were frequently too vague to be measured. Candidates almost universally state ‘user friendliness’ as a goal without really understanding what this means, or how they will prove that they have attained this goal. A simple solution is to examine these goals to see what is actually “testable” and/or “provable”.

It is not possible to design a solution without knowing what it should actually do and this section should give concrete examples of solution behaviour with a wide variety of data: given x data, the solution will process the data and produce y result. Six to eight well-specified objectives ought to be sufficient for most candidates.

Criterion A3

Dossiers demonstrated a lot of confusion about what exactly constitutes a prototype and an initial design. Often an early screen layout was presented as an initial design. A significant minority of dossiers presented screen shots, clearly coming from the functioning program, as prototypes.

A prototype design should be a simple module diagram or a list very briefly indicating the main functions/screens to be designed.

It is possible to design and then code a prototype, using Java or any other convenient language, and this has the advantages of providing some interaction and giving further insights into the eventual solution. However, the code is best relegated to an appendix if the candidate feels the need to include it.

Good prototypes were also completed in presentation and drawing packages.

Needless to say, candidates who complete the program first and then do section A afterwards find it almost impossible to score well in this section.

Criterion B1

A significant selection of dossiers clearly showed that some teachers do not know what data structures are, awarding full marks for discussions of graphic elements such as buttons and window labels, for example.

Many candidates used data structures which were appropriate to the problem. The treatment of this criterion has not changed with only a few candidates including any discussion of why the data structures they had used were suitable. The most common structure used is an array of records.

Candidates at standard level should be able to describe the basic data structures they have to work with such as classes (data objects), arrays, and files, and earn full marks on this criterion.

Candidates should remember to use actual sample data from the problem domain to illustrate their chosen data structures.

Lack of diagrams showing how the data structures change during the program and a discussion of other alternative data structures were the weakest points in this criterion. Samples and illustrations of how these data structures would change during the execution were missing in most cases.

Criterion B2

There is still a lot of confusion about what constitutes an algorithm and some candidates continue to present Java, copied or slightly edited from their completed code, as examples of their algorithm design. Again, this is poor practice and cannot be defended as it teaches the candidate nothing.

Algorithms should be presented in sufficient detail so they can be coded from the description. There was further confusion as to the purpose of pre and post conditions and teachers are advised to study this topic in a little more depth. A detailed understanding is not required but the candidates should at least know that the conditions are “contracts” made by the programmer and not the user.

Few candidates presented explanations of sorting, searching or file i/o algorithms.

Criterion B3

Little understanding of ‘the connections to data structures and methods’ was demonstrated in many cases. Connections between modules was often apparent, but then it also needs to be made explicit which algorithms are in which modules and which data structures are manipulated for the top mark. Often this can be achieved by a sensible system of naming and by having an overall (structured) diagram accompanied by brief descriptions of each module—what it does and what it does it to. Obviously it is not necessary to repeat the algorithms and data structure details again so for a standard level dossier this may only require two pages.

This section should not be a summary of Classes included in the code or derived via JavaDoc comments or IDE generated module diagrams since these can only be done after the solution has been coded.

It is worth pointing out that the candidates do not need to revisit sections A and B because they changed their minds at coding time; as long as sections A and B are “complete, logical and useable” candidates can deviate from it in sections C and D. It is true, however, that this deviation may have an impact on subsequent criteria that depend upon the goals of section A2 (C2, C4 and D1). This impact will usually be small and letting those three (approximately) achievement levels go, will save candidates a lot of time in unnecessary back-tracking to the analysis and design stage.

Criterion C1

Code layouts were generally good with a broad range presented. Candidates should be reminded to pay attention to the formatting of code pasted into word-processed documents. Using landscape orientation for the code is recommended.

Mastery aspects should be commented on in the code as page numbers sometimes appear to change between indexing mastery and printing the final dossier. It is extremely frustrating for the moderators to have to search through long listings to find evidence of mastery. The page numbers must refer to the actual code listing and not to other sections, for example, the mastery index.

Criteria C2 and C3

These criteria were generally poorly documented. Many candidates provided a narrative account of usability and error-handling but failed to provide evidence in the form of references to code or hard copy.

There were a significant number of dossiers where error handling was considered to be user friendliness.

Criterion C4

If candidates choose to use D1 as evidence for C4 then that should at least be mentioned in the dossier. Too few candidates established a relationship between criterion A2 and C4. If teachers make some notes, say on form 5/IACS, as to their view of the effectiveness of candidates' solutions that is also helpful to the moderation process and to providing good feedback.

Some teachers tend to give full marks to dossiers that do not include the minimum documentation to prove the program works (for example, no screen shots). The onus is on the candidate to provide proof, not on the moderator to spend hours searching for it in the code.

Criterion D1

The majority of dossiers presented screen shots for 'hard copy' evidence, which, it should be emphasized, are vastly superior to text dumps. Candidates ignored or struggled with providing evidence for File I/O operations or sorts.

The best sections included tables showing what was being tested in the various screenshots, demonstrating a systematic approach by the candidate. Once again, it was common to provide more abnormal/invalid data testing than actual shots which demonstrate the solution works as claimed. Candidates should provide at least one run showing that features such as adding, editing, sorting, saving and deleting records actually work.

Criterion D2

Most candidates discussed possible improvements and effectiveness. More successful candidates also presented discussions of efficiency.

Criterion D3

The user documentation was well written by many candidates. However, installation instructions were often not given. Illustrations are essential for this section.

Recommendations for the teaching of future candidates

With the analysis sections showing some signs of improvement it is recommended that teachers and students spend more time understanding the purpose of criteria B1 to B3. The emphasis in this section should be on documenting their problem-solving strategies, developing a logical approach and demonstrating their thinking. This would probably benefit candidates by allowing them to develop better code solutions and gain skills which are transferable to other areas.

The selection of a task should be done in a way that provides enough challenge to keep candidates' interest and yet is achievable. The candidates need to be constantly in touch with the teacher and the client to understand and analyze the problem at hand. A thorough investigation will lead to a good, workable, efficient and effective solution.

Data structures should be chosen which will satisfy the minimum mastery aspects suggested.

In some cases GUI programming, often using code-generating IDEs, appears to be getting in the way of producing good dossiers. Many of these dossiers are full of event-driven graphical objects with little purpose and which make little contribution to demonstrating mastery but serve to clutter the code and obscure the work the candidate has actually produced—especially when presented in the hard copy section. The password dialogue is a case in point: no practical value is served by these in the majority of student dossiers. Text-based solutions which suddenly pop up error message windows are another pointless feature increasingly encountered in dossiers, unfortunately.

Teachers and students should understand that mastery is not simply the use of a required mastery aspect in code. A standard sort is not evidence of mastery of sorting, loops, selection or anything else without documentation demonstrating the candidate's understanding of the algorithm and hard copy evidence that it actually works.

Problems continue to arise from the granting of mastery for:

- sorting, nested loops, parameters and return values for undocumented code such as the ever popular bubble sort;
- methods with parameters and/or return values for the creation of a data object with its associated basic accessors and mutators, most of which are not used within the program;
- arrays when ArrayList is used, or the arrays declared are not used in a way that demonstrates the candidate's understanding of the manipulation of data stored in arrays.

In addition, mastery aspects were all too often poorly documented in stages A and B.

Further comments

Teachers and candidates should take every care to check that page numbers given on form 5/PDCS do actually correspond to page numbers in the code listing. Where teachers do not tick these boxes and yet the code shows apparent mastery of aspects, the moderator will generally not award the aspect, assuming that the teacher has his or her own reasons for not crediting the aspect to the candidate (eg the code in question was developed in class and never modified or understood by the candidate). Of course, it would help moderators if teachers remembered to make notes to this effect on the work or on the 5/IACS form.

A really helpful dossier would have the "mastery index" make reference to the code listing and the page where the use of the aspect is documented (in section B).

Overall, the standard of work was a little higher this session than last and quite a few schools are turning in high-scoring dossiers of around 60 well-written pages. Some schools continue to submit dossiers filled with excessive material (code, hard copy or JavaDoc are the main culprits) which add to the burden of students without contributing marks. Some of these dossiers exceed 200 pages which is a sad waste of a student's time in many cases.

The examiner would like to make a special plea to those teachers, coming in to the subject without a solid computer science background, to consider acquiring the necessary skills as a matter of urgency. Too many teachers do not have sufficient background to understand the mastery factors themselves and appear content to rest on the candidates' judgement.

Higher level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 15	16 - 31	32 - 40	41 - 50	51 - 61	62 - 71	72 - 100

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

Section A

Question 1

Well done by students.

Question 2

Well done by students.

Question 3

Many students did state that the primary key was needed to uniquely identify a row in a database table or record in a file.

Question 4

Not well answered. Files which are too large to fit in the available RAM need to use an external sort.

Question 5

Well done, but students need to be reminded that “Outline” requires more than just stating the name of a step.

Question 6

Well done, but as with Q5, students needed to provide a brief statement about the example rather than simply naming the processing method. For example,

> Real time: hospital monitoring devices connected to a patient that responds directly to inputs.

Question 7

Well done.

Question 9

Well done.

Question 10

Well done.

Question 11

Well done.

Question 12

Not well done. Transfer time = seek + latency + speed of total transfer. Factors like disk speed are too vague.

Question 13

Well done.

Question 14

Reasonably well done, students should note that this style of question links (b) with (a) and marks cannot be awarded otherwise.

Question 15

Many students stated that defragmentation software deleted unwanted disk blocks, rather than stating that it placed blocks of the same file side by side to improve performance.

SECTION B

Question 16

Well done in general. Students should continue to expect to perform algorithm traces.

Question 17

Students need to state clearly what they are representing by a 0 or 1. A major mistake was to confuse the S not buckled (0) with S buckled (1). Many students showed good understanding of how to minimise.

Question 18

Many students showed a good grasp of how to construct a systems flow chart. File access is either direct, sequential or indexed. The distinction between these was not clearly understood when linked with using an ID. The implication is direct access or possibly indexed access. File organisation is either ordered or unordered (random). Again students did not show a clear understanding of when one organisation is to be preferred. To keep records ordered requires constant sorting on disk, keeping records unordered without an index generally only allows sequential access. In this case (c) sequential organisation would be the simplest. (Note that serial v's sequential is no longer a distinction drawn in the course.)

Question 19

Part (a) was reasonably well done.

Parts (b) and (e) required the Java statements (see markscheme). Many students described how the methods would work. Students at higher level are expected to know how to use object oriented techniques.

Parts (c) and (d) were reasonably well done.

Teachers need to ensure that students at higher level are taught object oriented theory as described in the course.

Question 20

a) The diagram poorly drawn by a number of students. It is expected that higher level students can draw a basic block diagram of a CPU.

b) This was not well done. PC, program counter, holds address of the next instruction.

c) Reasonably well done.

d) Reasonably well done.

Question 21

a) This was clearly understood by students.

b) This part was well done.

c) Soundly done. Many students noted the need to automate the update process, but did not address the need to educate users on an ongoing basis.

d) The concept of a multi-user operating system was not well understood. Students need to be aware of the different capabilities: single user, multi-user, multi tasking per user and multi-threading per user application. (Note that threading is not a specific requirement of the course and would not be directly assessed.)

e) This was not well addressed. Higher level students are expected to be able to explain at a conceptual level that file access is linked to different access levels.

Higher level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 17	18 - 34	35 - 42	43 - 53	54 - 65	66 - 76	77 - 100

General comments

Most candidates were able to demonstrate their skill and knowledge in this examination. Many candidates performed reasonably well. There were some excellent algorithms constructed and many candidates gained a lot of marks on the Case Study question. However, some candidates did not answer sufficient questions. Many answers were vague, lacking depth and substance, because candidates did not posses good examination skills.

Most candidates managed their time well and there was no evidence that there was shortage of time.

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

Questions 1 and 2 were based on programming concepts and students either performed very well or poorly. Static and dynamic implementation of abstract data structures is taught to candidates but completing algorithms and code using programming languages was difficult for many of them.

Question 3 tested candidates' knowledge in file handling. Candidates either performed very well or poorly. Those who performed poorly were the ones did not have sufficient knowledge about files and file organization.

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

There is a wide knowledge base of candidates appearing for this examination. The syllabus seems to be covered by most schools and most candidates were able to attempt all questions. There were a few students who were either outstanding or very poor in their performance.

There were some excellent algorithms produced but also some candidates left out all of these types of questions.

For non programming questions, many candidates wrote answers that were very brief and tended to neglect the amount of marks assigned to each question.

The Case Study was evidently well known to the candidates and the level of response to this question was satisfactory.

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

Question 1

Parts (a) and (b) were well answered. Most, but not all, candidates who attempted these questions were able to state the number of data values that could be stored in the array and were able to outline the problem that could occur with the algorithm.

In part (c) many candidates lost marks because they returned the smallest value instead of the position of the smallest value in the array.

Many candidates attempted part (e) correctly. Some candidates tried popping an element off but failed to shuffle all elements after popping off and forgot to return the popped value.

Part (g) was difficult for many candidates. Most candidates suggested a dynamic implementation of the stack to improve efficiency and failed to provide suggestions for array based solutions. In part (f) some candidates were not able to state the BigO efficiency of the pop operation.

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

Question 2

Many candidates answered part (a) and they achieved some marks on this question. Some candidates failed to give examples or a clear explanation of a constructor.

Candidates demonstrated a good knowledge of dynamic data structures in part (b). Although the labelled diagrams of linked lists were provided, the standard of diagrams provided varied in clarity. Some candidates either attempted a diagram, but showed a reverse illustration, or did not label the diagram at all.

Some candidates misunderstood parts (b) and (c) and gave very similar answers to both.

Part (d) required candidates to construct an algorithm using an iterative or a recursive method. Many candidates did not achieve full marks for this question. Either their code did not include a check for a Null or for a “*”, or there was no return value.

Part (e) was the most successfully answered. Most candidates understood the difference between a singly and a doubly lined list. However, many lost marks because their answers were too brief.

Question 3

This question was attempted by most of the candidates.

Some candidates who had insufficient knowledge about files and file organization scored poorly in this question. Some answers to part (a) and part (b) were not very clear and some candidates explained abstract data structures like trees, arrays, linked lists, etc. For this reason, they performed poorly in this part and generally in all other parts because this question focused on the concept of file organization.

Some candidates were not sure whether the organization used was partially indexed or fully indexed. Some identified it correctly but then explained the search for a record incorrectly.

In part (c) when changes to the index or the file were suggested, candidates failed to describe the efficiency of the process, but many could explain in part (d) how the file organization could be changed to make inserting records more efficient. This suggests that students have not thoroughly understood the concepts of file handling.

In part (e) most candidates described the binary search correctly.

Part (f) was answered correctly by many students. They easily identified the reason for not being able to apply indexing to tape storage systems.

Question 4

This question was based on the Case Study and candidates had enough time to prepare for it. It was the most successfully attempted question.

Most candidates successfully answered part (a) and part (b).

In part (c), many candidates did not understand the question well. They could not explain the advantage of involving the intended users during the prototyping stage and some even assumed it to be the final product.

In part (d), candidates were asked to identify a reliable data entry method. Most candidates attempted the first part but completely forgot to address the issue of reliability.

In part (e), candidates were required to identify and describe two data collection methods during the analysis stage from a hearing impaired staff. Most candidates identified methods but failed to describe and compare the suggested methods.

Part (f) was successfully attempted by almost all candidates.

In part (g), in order to achieve full marks candidates needed to identify a disabled operator and explain how the prototype would help the operator, which many did not suggest.

In parts (h) and (i), most candidates performed well. However, some candidates gave vague, general and ambiguous answers, and some lost marks as they either failed to explain the advantage or suggested very similar situations.

Recommendations and guidance for the teaching of future candidates

Teachers are advised to:

- distribute a copy of the syllabus to each candidate at the beginning of the course, then they are able to ask about topics they find difficult;
- explain the action verbs and how they relate to examination questions so that candidates can focus on exactly what is required;
- counsel students to concentrate on answering questions as asked and not to waste time writing down everything they know:

- expose candidates, especially the weaker ones, to programming concepts as they need to develop their confidence in understanding and writing programs: trace tables are a good way of improving programming logic and should be incorporated into day-to-day classroom teaching;
- provide candidates with as much examination practice as possible to improve their examination techniques: answering questions from past IB examination papers will allow for the reinforcement of subject matter and provide familiarity with the structure and content of different types of questions.

Standard level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 11	12 – 23	24 - 32	33 - 37	38 - 43	44 - 48	49 - 70

General comments

Most students coped well with the standard of the paper and there was no evidence that students had too little time to attempt all questions.

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

Students found question 14 difficult indicating that file handling was not well understood. Students need to think carefully about the basic logic needed to handle lists of data to derive information from the list. Lists are often processed using a linear search comparing items to some search value, which was the emphasis with this question.

There were no other aspects that showed widespread misunderstandings.

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

Other than list processing, students appeared well prepared across all aspects of the course. In particular, many students were able to tackle the algorithm trace with confidence.

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

Question 1

This was reasonably well done. A prototype is a model with limited functionality.

Question 2

Students showed sound understanding of the functions of an operating system.

Question 3

Many students did not match their answer to (b) with the run-time error mentioned in (a).

Question 4

Students showed a good understanding of data security.

Question 5

This was well done.

Question 6

Many students did not know what two’s complement is.

> +7 = 000111
>
> Flip bits = 111000
>
> Add 1 = 111001

Question 7

Many students answered this appropriately, part (b) in particular. It is important for students to appreciate that analogue data is continuous and digital data is discrete. Sampling is thus required of the continuous variable, each sample is recorded but what is between each sample is lost.

Question 8

This was well done.

Question 9

This question was not well done. Many diagrams listed components but they were not organised in a sensible manner.

Question 10

This was reasonably well done. Many students stated the need to have test data + expected results which are then compared to actual results.

Question 11

Again, this was well done: the index provides direct access without need to reference other elements in the list.

Question 12

This question was well done.

Question 13

The trace was, on the whole, performed better than in the past.

Question 14

This was not well answered. Many students showed limited understanding of the way in which data structures can be accessed.

a) The inputted code is compared to each element of the codes array, if no match then the code is invalid.

b) The customer's ID is used to search the complaints file until a match is found, this represents the first complaint for that customer.

c) Saves time via direct access but would need an index to be maintained.

d) By linearly looking down the prodID of the complaints file and totalling the number of matches the number of complaints for a product can be calculated.

Question 15

This was reasonably well answered. Students do need to ensure they provide sufficient detail in answers, in general, n marks = n points in the answer.

Recommendations and guidance for the teaching of future candidates

Some students seems to not read the question carefully and teachers should continue their efforts to equip students with effective question reading strategies. A simple technique is to try to prevent a 'brain dump' occurring. Students need to read and allow recall to occur, but before answering they are advised to pause, re-read the question and selectively addressing the specifics of the question.

List processing and file handling are important areas of computer science. The nature of the problems can be easily related to everyday events, e.g. searching for a page in a student's notebook. Teachers should look to relate the theory to practical everyday examples in order to arrive at the logic required and improve students understanding of the algorithm process required to search and access data in a list or file.

Standard level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 12	13 – 25	26 - 31	32 - 38	39 - 45	46 - 52	53 - 70

General comments

Overall the paper seemed a fair test for the candidates. Although some questions were more difficult than others, all parts seemed accessible to the more able candidates.

The paper seemed to have been clearly written. The only part that was misunderstood by some was Question 2(a), where some candidates focused on explaining the loop itself rather than the terminating condition.

There did not seem to be any time problems as virtually all of the candidates were able to complete all three questions, although the better candidates (who spent more time on Questions 1 and 2) did not do as well as expected on the Case Study question. It could be that the weaker candidates were, in fact, at an advantage as they appeared to have more time at their disposal to browse through the Case Study material during the examination.

Syllabus coverage is always a problem with Paper 2, as it consists of only three questions. It is always problematic finding Case Study questions which relate to all parts of the syllabus.

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

The students were unsure of the difference between types of memory and the extended response questions, in general, appeared difficult for candidates.

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

The students dealt well with the different aspects of programming, which is a welcome change from previous years. They also dealt reasonable well with the various areas of the Case Study.

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

Question 1

There was a significant improvement on previous years in the candidates’ ability to deal with algorithms, as was seen in the marks gained on this question.

a) Students who had studied input devices answered well. Others dealt with how the algorithm worked, which was not what was asked.

b) This required the students to write the method that calculated the number of hours parked. Not only were many full marks awarded here, but there were also at least 14 variations that successfully produced the correct answer. Some of the variations were quite complex, and it is interesting to note how logical thinking varies between candidates.

c) & d) These parts dealt with substrings and parsing and were, again, handled better than in previous years.

e) Here the students were asked to identify a problem with a changed scenario and propose a solution to it. Many students correctly identified the potential problem, but not so many were able to come up with a clear solution.

Question 2

a) & b) These parts dealt with understanding and writing code, and again there were many good answers. Most were aware that “x<y” in part (a) was the terminating condition for the loop, but not all were able to explain it.

Most students then made a good attempt at re-writing the loop, but only very few included the initial condition.

c) & d) These parts dealt with theory and proved problematic for candidates, with many unsure of the roles of cache memory and virtual memory. Some confused the two. It may well be that schools are spending more time on algorithms at the expense of theory.

Students should be made aware that if a question has three marks allocated to it, then three separate points need to be made and students should try to answer accordingly.

Question 3

Some candidates who had made themselves familiar with the Case Study were able to pick up high marks here.

The two parts that were more difficult to answer were the extended answer parts (e) and (f) which required a more thoughtful approach.

Interestingly, the weaker candidates (with regard to Questions 1 and 2) seemed to do better on the Case Study question than the more able students, presumably because they had more time to search through the material during the examination.

Recommendations and guidance for the teaching of future candidates

Teachers should aim to provide opportunities for candidates to:

- get the balance right between studying theory and preparing for algorithms;
- recognize that the mark allocation for each question relates directly to the amount of information required;
- study the Case Study well in advance.