November 2015 subject reports

Computer Science

Overall grade boundaries

Higher level

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 16	17 – 32	33 – 44	45 – 53	54 – 63	64 – 72	73 – 100

Higher level internal assessment

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 7	8 – 14	15 – 20	21 – 24	25 – 29	30 – 33	34 – 40

General Comments

Generally the work submitted followed the instructions as laid out in the Computer Science Guide and the Handbook of Procedures for the Diploma Programme 2015, section B4.4.

Some recommendations from the Principal Moderator:

- The product folder must contain some evidence of the product in order to verify it exists – preferably both the final product and the product at design stage, for example both executable jar file and original java files, or an Access database client version (that opens a full-screen switchboard) and a design version that does not. In extreme cases, for example when the product is being developed on-line, the product folder should contain screenshots of the product being created.
- The video/screencast should be 5-7 minutes (maximum) and should only show the proper working of the final solution – the use of techniques should be described in criterion C using extended writing. It is suggested that candidates use their Criteria for Success (criterion A) and their test plan (criterion B) to script the screencast.
- Even though it is not a requirement, teachers are asked to provide pertinent comments on how they awarded marks to the candidates in their sample. This facilitates the moderator's validation of the teachers' marks.

The range and suitability of the work submitted

The described scenarios typically allowed for worthwhile projects. As expected, the majority of solutions concerned programming projects and the majority of those had been coded in Java. On the other hand, an encouraging number of candidates tried their hand at web design, (Access) databases and Android app design. It is hoped that the range of solutions continues to expand.

The quality of the solutions showed a wide range and not all solutions had been developed to the level of complexity expected of IB DP candidates. Some examples of trivial products include: Java programs that mainly focus on GUI and not on actual functionality, Java programs that consist of one class only, Java programs consisting of a Greenfoot template with only two methods overwritten, rudimentary versions of freely available games (like Sudoku), Access databases that contain just one or two tables or non-relational tables, websites that are template-based (Wordpress, Wix or Weebly) or that have minimal content, basic Excel projects, Scratch projects that had not been properly designed.

Highly successful solutions tended to incorporate features from more than one software. For example, website projects that incorporate JavaScript / PHP / SQL functionality, or programming projects that interact with an Access database or with on-line resources.

Candidate performance against each criterion

A Planning – This remains the most straight-forward criterion. However, several candidates did not follow the expected sequence:

- investigate a situation,
- identify client/adviser,
- explicitly consult the client (and/or adviser),
- describe the scenario with explicit reference to evidence of the consultation added in an appendix,
- choose a solution,
- describe the rationale for the solution and also for the software to be used,
- outline comprehensive Criteria for Success for the chosen solution.

Too many candidates decided on a product ('I want to make a website/program a game') and then found a client to match. Some schools adopted a standard approach to the report where all candidates used the same components, layout and content but applied to different problems, leading to generic reports. These approaches must be discouraged. Contrived tasks and clients were routinely seen in the weaker samples submitted. Too many candidates had generic success criteria – these criteria must be specific and testable. The Criteria for Success are essential to the project and must be explicitly addressed in the test plan (B) and in the evaluation (E) and preferably also in the screencast.

B Solution overview – Comparatively this was the worst addressed criterion, and candidates typically only provided an outline design or even screenshots from the final product (which were discounted). The structured approach of prototyping together with client feedback allowed some candidates to achieve a higher mark. Records of Tasks were generally only partially

complete, typically because the final product had not been implemented / fully tested by the client. A wide variety of test plans were seen. The better ones aligned with the Criteria for Success.

Please note: The use of the proper template in forms.zip is mandatory – the use of a different version, with incorrect columns and headings, will be penalized. If no Record of Tasks is included or if there is no evidence of a design then 0 marks will be awarded.

C Development – Most candidates made a good attempt to document the development of their product and the techniques used. However, the quality of the explanations and the completeness of techniques typically left something to be desired. The complexity of the product must be justified by the candidate in the write-up. A seemingly complex product without proper explanations of complex techniques used in the product, only achieves moderate complexity. Similarly, high ingenuity must be justified by algorithmic thinking (e.g. explanations of complex data structures, algorithms or macros).

D Functionality and extensibility of product – The screencast should only show the proper working of the solution as outlined by the Criteria for Success. Many screencasts focused instead on the development of the solution, which made them too lengthy. Others only showed the working of the interface, without showing actual functionality of the intended solution. Screencasts must be included in formats that are recognized by major video software like VLC player. There is no need to document extensibility in extended writing or in the screencast.

E Evaluation – The final product (after testing) is expected to be implemented and used/tested by the client before client feedback is given. For full marks evidence of feedback must be included (typically in the appendix) and it must be discussed and referred to in the candidate's evaluation against the Criteria for Success. Recommendations should be realistic in relation to the actual product – for example 'adding network capability' is not a realistic improvement for a low-level product.

Recommendations for the teaching of future candidates

The aim of the IA in Computer Science is to create a working solution for a real client. The consultation (included as an appendix) should be the basis for the description of the scenario, leading to Criteria for Success of a chosen solution. All high scoring projects showed ample evidence of client involvement.

Criterion B should provide evidence of a rigorous design stage with an overview of all five stages of the project (including the actual intended use of the product by the client) in the Record of Tasks, detailed layout design sketches that include annotations for complex techniques, evidence of algorithmic thinking (in the form of flowcharts, UML diagrams, pseudo-code, ER diagrams, structured database decomposition using NF, query and macro design), and a test plan that addresses all Criteria for Success. All high scoring projects included a thorough design stage.

Criterion C provides candidates with the opportunity to demonstrate their knowledge and understanding of the tools and techniques used in creating the product. The use of tools/techniques should be explained in relation to screenshots that show their use.

Criterion D does not require written documentation. The screencast should be limited to about 5 minutes and should only show the proper working of the final solution. The structure of the screencast should be scripted by the candidate. For example, the screencast could show the testing of the implemented solution following the test plan from criterion B. Successful screencasts showed comprehensive evidence of the solution's functionality with lots of data, but were edited to avoid viewing tedious data entry. Candidates are advised to test their screencasts on different media players and devices to ensure the playback is correct.

Extensibility is evidenced by a detailed design in criterion B, by a detailed description of the creation process in criterion C and, in case of a programming project, by a properly structured and annotated code listing in an appendix.

Criterion E should provide evidence of a rigorous evaluation stage. The client feedback (added in an appendix) should be discussed by candidates as part of their own evaluation of the solution. A table showing the Criteria for Success with a 'met/not met' evaluation is not sufficient to achieve in the highest descriptor. Recommendations for improvement should go beyond the success criteria that have not been met.

A word of caution: treating the project as a purely academic exercise typically means that there is no proper client and that the solution is not being implemented, which will have an impact on criteria A, D and E.

The recommended word count **for each section**, as indicated in the TSM, is only for guidance. The **overall** word count of 2000 words however, is a fixed limit and a moderator is not required to read beyond this limit, which could cause a loss in marks in criterion E.

Further comments

For additional information regarding the Computer Science IA, please consult:

- Computer Science Guide (pages 56-72).
- Teacher Support Material (Internal Assessment) available on the OCC.
- Forms.zip templates.
- Submission of the Computer Science IA in the Handbook for Procedures for the Diploma Programme 2015 (Section B4.4). Note that the Handbook is updated yearly.
- IB Coordinator Notes.

For additional professional development regarding the Computer Science IA, please consider:

- Getting involved in the Computer Science OCC discussion forum.
- Registering for Computer Science workshops (either face-to-face or online).

Higher level paper one

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 17	18 – 35	36 – 47	48 – 56	57 – 66	67 – 75	76 – 100

General comments

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

Altogether the impression is that candidates are exposed to a way of teaching (that consequently moulds the understanding of the subject and specific areas) that comes from teachers that might not be particularly expert in IT or computer science.

Most candidates show a generally satisfactory knowledge and understanding of very basic facts and notions, but systematically fail in entering into explanations or discussions of more technical aspects. This is a major problem, because it affects ample areas of the guide, spanning from basic programming, networks, system architecture, and security.

In several Objective 3 questions candidates have lost marks by simply going off course and attempting to build some unwanted answer about socio-economical considerations where the question was evidently targeting computational and technological concepts.

For example:

- Many candidates built a truth table for 3 inputs with any number of rows varying from 5 to 7: this is not just the logic to be incorrect, rather candidates clearly do not have a method and do not know that with 3 inputs one gets always and only 8 rows.
- Client-server is basic terminology in computer science, but many candidates think that 'client' is a customer wishing to buy something.
- Poor ability of writing code: while the basic loop construction is usually non-problematic, many candidates have failed building a more structured and articulated second exercise. Some have attempted the construction in a very confused way where loops had no indication at all of how many iterations where needed (or a condition); indexes were not always meaningful. Others have bypassed the code construction by providing an explanation in plain English of what the program had to do: this was anyway insufficient to gain marks, because the explanation did not address ANY of the computational features expected.
- Trees and pointers: apart a few cases, it seems that candidates understand the structure of a tree, but many do not have clear ideas about its nodes. In fact some

candidates have answered with three trees, one for all the key information, one for all the left pointers and one for all the right pointers, as if a tree were the superposition of three independent tree-structures, yet without giving any relation among the three trees.

Altogether, the amount of misconceptions seem to be quite high and broadly distributed in this session, at HL.

Unfortunately, the kind of flaws mentioned above are not concentrated just on those answers that eventually end up in the low scale of all scores: many of these misconception are present also in responses that end up above the average.

It is also worth mentioning that several of the major flaws could be easily prevented in the future, with more solid teaching methods and with a mind-set of teaching a scientific discipline full of technical and technological content that requires accurate answers.

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

Factual knowledge is generally well understood.

Any question that mildly addresses user interface/user machine interaction are also well understood, but not necessarily so the technical aspects needed to provide a technical inclined explanation.

Elementary programming is fine, but the more articulated one is generally not.

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

Question 1

In part (b) several candidates provided very generic answers without indicating ANY of the many technological solutions expected in the markscheme. Instead, they provided explanations of **why** a GUI could help or not, and whether it is right or not to do so.

Question 2

Many candidates know about tunnelling and have explained in good detail the benefits in **using** it, however **without explaining how it works.** Only a few candidates have provided **two** technologies; gateways are usually not known, and some candidates have addressed authentication mechanisms.

Question 3

Many candidates seem not to know the NOR, and have used an OR instead. Some candidates have used a **unary** NOR (as if it were a NOT). Worse, several candidates have returned a truth tables with 5, 6, 7 rows. Others have attempted an incremental construction of the truth

table, as if it were a database table, so that all their tables were just having 4 rows, and above all whatever was the output of the final composition, there was no direct link with the values for the inputs A, B, C. This entire area showed very little understanding by many.

Question 4

In part (a), many answers were too generic, but where possible, some partial marks were awarded using benefit of the doubt. In part (b), many irrelevant answers in the tone 'clients are the parents that pay for the school'. In part (c)(i) several candidates have answered with vandalism/theft/natural disasters… While these certainly occur the focus of the subject is in Computer Science, and there are many more technical reasons stemming from Computer Science that can provide an answer to this question. Teachers should please try to make any effort to steer this understanding in their cohorts of candidates. In part (c)(ii) again, many answers going off course, such as pay a security service, dogs, extra fences, secure storage, etc.

Question 5

Some candidates skipped this question, while many have just answered with 'this code doesn't do what it is supposed to have to do'. Some responses try to attempt an answer but with insufficient detail. A common problem in answers is the confusion between MOD and DIV. It seems that the majority of candidates have an inclination to 'fix' a program by 'adding' features (for example an extra loop) and almost never by 'removing' or just 'changing operation': this is a methodological problem. Candidates should be exposed during the course that 'adding stuff' is not the only way to solve problems.

Question 6

In part (b), in a few cases, the explanations were too generic; only describing how a queue/stack works. A few candidates incorrectly used 'queue/dequeue' operations also for the stacks.

Question 7

In parts (d)(i) and (d)(ii) there was generally correct understanding, but often the answers lacked meaningful detail. Saying that 'it allows many programs to run together' shows general understanding and is not enough to get full marks, for example. Multi-tasking and multi-user are treated the same way by a few candidates.

Question 8

In parts (a) and (b), many candidates answered correctly, but sometimes creative answers were seen in part (b), including fractions. In part (c) marks were lost in answers that only expanded the acronym MAR and MDR. In parts (e)(i),(ii) & (iii), drawings in (i) showed general understating with some faults; the connections among different parts was not always correct.

Question 9

In part (b) while there is general understanding of the effects of an interrupt, the description in many answers was far too elementary and non-specific. Technical terminology is often not used.

Question 10

In part (a), apart from a few very creative answers (up to three overlapping trees or just one long zig-zag tree), most of the candidates that attempted it got high marks, getting more or less a correct answer without addressing really full details on the nodes. However, the lack of precision, and the lack of communicating precision while designing a data structure is something that should be addressed, because it is at the base of any computational activity, including programming. In other words, the unspecific answers gathered in this question are an indication that if candidates cannot produce a data structure with all its relations as a template, then there should be no surprise that they cannot even program well. Part (b) was a comparable situation to the previous one; some candidates do not draw a node with a field for the pointer, and pointers just originate in a sketch seemingly out of visual memory, and not by reasoning in terms of data structures. Computational understanding does not seem well supported. In part (c)(ii) the majority of responses seemed to not consider storage requirements in terms of space required for pointers. Part (d) was generally correct, apart from a few exceptions. Very often the presentation was confused.

Question 11

In part (a) responses were generally correct, using a loop construction. A few answered with the sum of the elements in the array, which is unfortunate to see at HL. In part (b)(ii) many made the calculation wrong, and as it was only 1 mark, the result was either correct or incorrect. There was no “benefit of the doubt”. In part (b)(iii) some candidates did not consider the influence of people that, from the region, move elsewhere. In part (c) a few candidates got it right; some that made a serious attempt got high marks despite some flaws (including boundaries on loops), showing that at least they have a general understanding of the logic and can master the basic code constructions. But the majority of the answers did not show clear ideas at all. Handling the two indexes simultaneously seems a source of difficulty; formulating the appropriate boundaries for the two nested loops is also a problem; assignments are therefore often incorrect. A common mistake was the most external loop not being based on 10 years.

Recommendations and guidance for the teaching of future candidates

Ensure the subject is taught well, by a teacher with a genuine competence in computer science, with a method, and by addressing the whole subject from a technical and scientific point of view (not just from the perspective of the user).

Teach how to structure answers, how to read and understand what is expected from the question. Teach systematically and structurally: data structures should be manipulated well, and these are at the basis of any programming activity.

Higher level paper two

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 10	11 – 20	21 – 27	28 – 31	32 – 36	37 – 40	41 – 65

General comments

The overwhelming majority of the cohort attempted Option D, Object-oriented programming. Therefore, this feedback will mainly concern the OOP questions. Option A, Databases, was the second most popular option.

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

Algorithmic thinking and coding, particularly when there were relationships between objects.

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

The principles of inheritance seemed well understood. Basic topics such as the use of constructers and encapsulation seemed well understood.

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

Question 14

The answers on the constructor and additional variables were answered well, but the two coding questions were not. Candidates often mixed up methods and variables, and not all used the accessor methods. Few made use of the toString() method in part (e).

Question 15

Very few responses dealt with the actual design of the classes (part b) in order to provide confidentiality. Most gave answers that focused on the use of the programs. The UML design question was quite well answered. The question on extended character sets showed clearly that some schools had not covered this topic.

Question 16

This algorithm question was poorly answered and in general the coding questions showed a fundamental weakness in this essential area of the course.

Question 17

The UML diagrams were constructed well with most candidates understanding the basic principles of inheritance. Modularity was also well-explained.

Question 18

The responses show a general understanding of the functions of a stack, but once again weakness was shown at the algorithmic level (with the trace question in part (c) and the improvement to the method in part (e)).

Question 19

Quite a few candidates did not attempt the recursion question. However, those who did generally answered well.

Recommendations and guidance for the teaching of future candidates

Object-oriented programming

There is a considerable range of ability shown on the November papers which clearly highlights the different level of treatment that is given to algorithmic thinking and its relationship to OOP design. This is a fundamental area of the course, particularly for candidates opting for this option, as it forms a fundamental part of both Papers 1 and 2. Considerable time must be devoted both to algorithmic thinking in general and to its application to OOP. In order to approach this paper with confidence candidates need to be comfortable with working with programs in which objects have relationships with others (e.g. when an attribute of one object is an object of a different class). Familiarity must also be shown with using the accessor and mutator methods and a clear understanding of the reasons for their use would help this.

UML diagrams are used to provide visual information about both the individual classes and the relationships between them. This information should be as complete as possible. Although the questions on this topic were answered quite well, the use of +/- for public/private and the means of showing relationships (“use of”, “has a”, “is a”) and the corresponding types of arrows linking classes is still not completely understood.

The topic of recursion is one which continues to cause difficulty. However, once the basic understanding is gained then the actual questions at this level are not too difficult. Some schools clearly need to allocate more time to this topic. Establishing a clear and consistent method of tracing recursive algorithms and providing extensive examples of this would help.

Databases

Only a few candidates took this option. Questions on normalisation will always appear on this paper and will tend to be one of the difficult topics, so considerable practice both at carrying out the normalisation process (going through the different levels) is essential. The starting point is always to carefully establish the field or fields that will become the primary key and uniquely identify each record (row in the table). In some cases in may be necessary to devise a new field for this purpose. The consequences from a failure to normalise a database must be clearly understood.

Higher level paper three

Component grade boundaries

Grade:	1	2	3	4	5	6	7
Mark range:	0 – 4	5 – 8	9 – 11	12 – 14	15 – 17	18 – 20	21 – 30

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

The depth of understanding required in this exam – particularly for questions 3 and 4 – is not being shown by many candidates

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

The earlier, definition-type, questions were quite well-answered.

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

Question 1

This was answered well. Most schools are aware that this question will require a definition or a basic description of one of the terms in the Additional Terminology. The most minimal revision should include sufficient knowledge of these terms.

Question 2

Most candidates were familiar with the concept of push and pull. The process of using QR codes was not so well described. There are two possible applications currently used: one in

which the ATM displays the QR code and the phone scans it, and one in which this process is the other way around. The fact that authentication with the bank must somehow take place was not well-understood.

Question 3

This was not well-answered. This (and question 4) require a detailed understanding of the underlying technologies, and it is clear that not all schools are going past the descriptive level when dealing with the case study. SSL/TLS is a process that is fundamental to the security of transactions on the web and should be well-understood by all candidates.

Question 4

A mainly descriptive response will not pass the adequate (4-6) level on this paper, and to reach the proficient level (10-12) candidates need to show a sound understanding of the technical aspects of biometrics and the consequences of their use. The aspects that needed to be addressed here were: the technology behind various biometric methods, the advantages/disadvantages of their use with regard to security during the authentication process, any usability issues and a comparison with other (non-biometric) methods. This clearly requires a comprehensive understanding of the case study which was only shown by a few of the very best candidates.

Recommendations and guidance for the teaching of future candidates

It is clear that some schools are addressing the case study component in a systematic way that involves a 12 month programme. Each case study will be available at least 12 months before the first exam and the similar structure of the different case studies means that a similar approach can be taken each year. However, the responses from many schools suggest that improvements can be made in the way they treat this component. In addition to the in-school activities, the nature of the case study requires a significant degree of independent research on the part of the candidate in order that they confidently approach the 12-mark question with the level of understanding necessary to achieve the highest mark band.