CARIBBEAN EXAMINATIONS COUNCIL

REPORT ON CANDIDATES' WORK IN THE CARIBBEAN ADVANCED PROFICIENCY EXAMINATION MAY/JUNE 2010

COMPUTER SCIENCE

GENERAL COMMENTS

This was the second year for which the revised syllabus was followed. There were three examination papers in each of Units 1 and 2, namely, Paper 01, Paper 02 and Paper 03. In each unit, Paper 01 and Paper 02 were examined externally by CXC while Paper 03, the Internal Assessment, was examined by teachers and moderated by CXC.

In each unit, Paper 01 consisted of multiple-choice questions that were designed to test candidates' breadth of coverage of the syllabus. Paper 02 consisted of essay-type questions that were designed to test candidates' depth of understanding of the syllabus. Thus, candidates were expected to show deeper insight and understanding of the topics examined in Paper 02.

The individual contributions of Paper 01, Paper 02, and Paper 03 to the final grade are 30 per cent, 50 per cent, and 20 per cent, respectively.

In Unit 1, 73 per cent of the candidates obtained Grades I–V. In Unit 2, 92 per cent of the candidates obtained Grades I–V. Performance on the School Based Assessment for Unit 1 was roughly the same as in 2009. However, performance on the School Based Assessment for Unit 2 increased by more than 10 per cent compared with 2009.

Performance on questions involving computing programming continues to be poor; however, there have been small improvements in certain types of programming questions. It is clear that many candidates are still not getting the experience they need in writing and testing real computer programs on their own.

Of particular concern is the fact that performance on other types of questions such as those on Computer Architecture and Organization has also been very poor. This indicates a lack of adequate preparation on all aspects of the syllabus.

DETAILED COMMENTS

UNIT 1

Paper 01 – Multiple Choice

Performance on the 45 multiple-choice items on this paper produced a mean of 51 out of 90 with scores ranging from 16 to 86.

Paper 02 – Essay Questions

Section A – Computer Architecture and Organization

Question 1

This question examined candidates' knowledge of logic gates, truth tables, binary counters, line decoders, and the internal representation of data on a computer.

Overall, the question was poorly done. It is clear that candidates did not master the basics of computer architecture and organization. Some candidates ignored Part (d) which involved the representation of data on a computer. Several others did not attempt the questions on truth tables. This is cause for concern since the representation of data on a computer and truth tables are the most basic and straightforward items in the syllabus.

In Part (a) of the question, most of the candidates correctly identified the logic gates and gave the required truth tables. However, some candidates mixed up the gates or identified the gates as NAND and/or NOR gates.

Part (b) was poorly done by most candidates. Of the candidates who attempted the question, about 98 per cent of the responses were incorrect. It should be noted that the binary counter goes through a prescribed set of states (0–15) upon application of an input pulse such as a clock pulse. After the counter reaches 15, it goes back to 0.

A fair attempt was made at Part (c) (i). However, most candidates did not label the block diagram properly. In Part (c) (ii), candidates demonstrated a clear lack of knowledge of the truth table of a 2-to-4 line decoder. Part (c) (iii) was also poorly done. Most candidates did not attempt this part. Of those who attempted this part, the majority of them gave incorrect responses.

Part (d) (i) was fairly well done. In Part (d) (ii), most candidates were able to perform the calculation but several of them had difficulty in explaining why the result of the calculation could be stored as a 4-bit binary number. Most candidates attempted Part (d) (iii); however, their calculations were incorrect resulting in incorrect responses. Part (d) (iv) was well done indicating that candidates have a very good grasp of two's complement representation.

Question 2

This question tested candidates' knowledge of storage devices, cache memory, instruction set, instruction formats and instruction cycle. It was misinterpreted by most candidates and answers were vague.

In Part (a), most candidates gave qualitative responses such as *moderate*, *low* and *high* when comparing the capacities of ROM, RAM, hard disk and CD-RW. Similar responses were given when comparing the access speeds of the devices. Very few candidates gave a quantitative response in terms of numeric capacities and placing them in order (ascending/descending). Also, instead of comparing the devices based on access *speed*, many candidates compared the devices based on access *method*.

Part (b) was poorly done. Most candidates were unable to explain how cache memory works and to correctly describe one benefit of its use. In Part (c), candidates focused mostly on the activities of the instruction cycle. Most of them were unable to explain what the instruction set of a CPU is, and to describe types of instructions and typical instruction formats.

Section B – Problem Solving with Computers

Question 3

This question tested candidates' knowledge of the basic control structures used in computer programming and the use of these control structures in a given algorithm. The question also examined candidates' knowledge of representing algorithms using graphical techniques such as flow charts and their ability to understand and modify an algorithm based on given specifications. Overall, the question was not done very well and it appeared that many candidates misinterpreted what was required.

In Part (a), some candidates were unable to identify the three basic control structures of sequence, selection and repetition. Some candidates identified them as input, output and processing. This is cause for concern since the three control structures are the building blocks of computer programs which account for roughly half of the Computer Science syllabus. Most of the candidates misunderstood Part (b) and attempted to explain the control structures without making reference to the algorithm given, as instructed in the question.

In Part (c), some candidates interpreted graphical representation to mean drawing images of actual bananas, pineapples etc. It was expected that candidates would draw a suitably labelled flow chart corresponding exactly to the algorithm specified. One problem with the flow charts given was that the flow lines between components were often missing. In some instances, incorrect symbols were used. Many candidates wrote *for j = 1 to 100* in the decision box, which was incorrect; it should have been, is j equal to 100? The loop is achieved by an arrow returning to a previous point in the flow chart.

The responses that were given in Part (d) demonstrated that candidates did not have a very good understanding of tracing through the logic of a fairly straightforward algorithm expressed in pseudocode. As a result, they were not able to make the required modifications to the algorithm resulting in this part being done badly by most candidates.

Question 4

This question tested candidates' ability to trace through a given algorithm to determine its output and to develop an algorithm from a given specification. The question also examined candidates' knowledge of the *Implementation and Review* stage of the problem solving process. Overall, this question was poorly done.

In Part (a), candidates were able to trace through most of the algorithm. However, many of them were unable to produce the correct diagram of an arrow pointing upwards. In general, Part (b) was done fairly well. Most of the candidates who attempted the question scored at least three marks. This is a good sign since it indicates that candidates' problem-solving abilities are improving.

In Part (c), it was clear that many candidates did not read the question thoroughly. They focused on the actual question and ignored the preamble. Hence, they did not answer the question within the context of the video club. As a result, many of the candidates achieved low scores on this part. Also, some candidates described the entire problem-solving process which was not required.

Section C – Programming

Question 5

This question tested candidates' ability to take a specification of a program and to write a C program which achieves the desired functionality. It also examined candidates' knowledge of programming language features and the appropriateness of programming languages for different types of applications.

Part (a) of the question was generally answered poorly by candidates. Most of their responses described types of programming languages and the generations of programming languages. Few candidates gave the correct response that *mobile devices have limited graphical capabilities and processing power so the programming language will offer less graphical features and smaller libraries.*

Part (b) was fairly well done. Most of the candidates were able to correctly differentiate between a *character* variable and a *string* variable. Part (c) was poorly done by most of the candidates. Many of the responses demonstrated poor programming skills and an inability to manipulate files in the C programming language. Some candidates attempted to answer the question using arrays. However, this was unnecessary since the data for each employee could be read and processed in a *while* loop without saving the data on all the employees.

Question 6

This question tested candidates' ability to write functions in the C programming language. It also examined their understanding of what constitutes good programming style.

In Part (a), the majority of candidates were able to correctly identify spacing and indentation as two ways of improving programming style. The use of consistent case (for example, lowercase) when naming variables is another way to improve programming style.

For Part (b), most candidates were able to explain the purpose of a *struct* in C. However, in many of the responses, the idea of a record was not mentioned.

Responses to Part (c) demonstrated a clear lack of knowledge of modular programming in C using functions. Many candidates did not write the function prototype correctly (return type, name of function, followed by parameter list). The code given in their responses often did not have return statements and many of them were not able to call the functions properly; some candidates even specified the types of the variables when making the function call.

UNIT 2

Paper 01 – Multiple Choice

Performance on the 45 multiple choice items on this paper produced a mean of 54 out of 90 with scores ranging from 22 to 84.

Paper 02 – Essay Questions

Section A – Data Structures

Question 1

This question tested candidates' knowledge of abstract data types and their implementation. It also examined candidates' knowledge of manipulating abstract data types to achieve a desired outcome. Most candidates attempted this question. However, the quality of the responses was generally poor.

In Part (a), most candidates had a general idea of what an abstract data type (ADT) is but most of them could not define it as a specification of a set of data and the operations that can be performed on the data.

In Part (b) (i), some candidates confused the singly linked list ADT with other abstract data types such as a queue or a stack. The ADT operation could have been insertFirst (LinkedList, data) or insertLast (LinkedList, data) or something similar. In Part (b) (ii), most candidates responded well. However, a few candidates omitted the pointers to indicate the beginning and end of the list.

In Part (c) (i), most candidates did not realize that the computer storage they were asked to describe was simply the data structures associated with the implementation of a queue (an array and several integer variables to keep track of the beginning and end of the list).

Part (c) (ii) was generally well done. Part (d) was attempted by most of the candidates. However, some of the candidates did not demonstrate the skills required to reverse the elements of the queue using the stack for temporary storage.

In Part (e), most candidates described the attributes of either a stack or a queue. However, few candidates were able to explain that a stack can contain an embedded linked list and thus, operations such as pop() can be implemented by calling the linked list operation, deleteFront().

Question 2

This question tested candidates' knowledge of sorting and searching algorithms. It was generally answered poorly by most candidates. However, there were very good responses from a few candidates.

In Part (a) (i), most candidates did not give a detailed description of how the selection sort algorithm works. It was important to specify how the minimum element from 0 to 9 would first be found and swapped with the element at position 0. It was also important to specify that succeeding iterations would go from 1 to 9, 2 to 9 and so on, until no more elements are out of place, that is, when the 8 element has been put in its correct position.

In Part (a) (ii), most candidates skipped the second pass in which no swap was made. However, this is a valid pass even though the array remains the same as it was at the end of the first pass.

The responses to Part (b) were of a poor quality. The question required candidates to write a program that performs a linear search. Many candidates incorrectly put the 'if not found' test within the *for* loop,

instead of the outside. This would cause the message to be printed every time the element in the array is different from *target*.

In Part (c), explanations of binary search were given but this was not required. Candidates were expected to answer that the binary search should be used when the elements of the array are sorted in ascending or descending order.

Section B – Software Engineering

Question 3

This question tested candidates' understanding of various software engineering concepts as well as their ability to draw data flow diagrams corresponding to a given narrative. It was done fairly well by most candidates.

In Part (a), many candidates were able to correctly discuss the attributes of well-engineered software. However, some of them were unable to list the names of the attributes correctly, for example, reliability was often used instead of dependability as one of the attributes. Also, many candidates listed portability as an attribute but were unable to say what it was.

Most of the candidates correctly discussed the need for user involvement in the software development process in Part (b). However, several of them failed to differentiate between the role of users and managers in the development process. In many cases, the manager's role was simply omitted.

In Part (c), many candidates were not able to clearly explain why a software system needed to be upgraded. The expected response was that *the needs of a business change all the time since the business environment is changing and thus the software must be able to adapt to accommodate these changes.*

In Part (d), a few candidates did not know which symbols to use when drawing a data flow diagram. At times, entity-relationship diagram symbols were used. Also, several candidates gave a context diagram instead of the Level-0 diagram requested.

Question 4

This question tested candidates' knowledge of the feasibility study and the requirements specification document in software engineering. It also tested candidates' ability to draw an entity-relationship diagram from a narrative and to specify ways in which a piece of program code can be tested. The question was well-attempted. However, it was poorly answered.

In Part (a), most candidates identified economic and technical feasibility as reasons for undertaking the feasibility study. Few candidates recognized that it may be possible for user needs to be satisfied with existing software and hardware.

Many candidates were able to gain most of their marks in Part (b) of the question. However, several of them omitted the identification of the primary key, for example, underlining the *employee number in* some notations.

Many candidates did not seem to have read Part (c) of the question properly and so did not relate their responses to the parameters of the function *lsearch*. Many candidates simply described general forms of testing and did not apply the principles of testing to the function given. For example, one test is to call *lsearch* with an array of integers which does not contain the given *key*; if *lsearch* is working correctly, it should return -1.

Section C – Operating Systems and Computer Networks

Question 5

This question tested candidates' knowledge of the OSI model for computer communication and various operating system concepts such as page fault and process states. It also tested candidates' knowledge of GPRS. It was generally poorly attempted by most candidates, most of whom obtained under 10 marks, with a significant number obtaining 0–5 marks.

In Part (a), most candidates were able to correctly identify the seven layers of the OSI level. However, a few candidates mixed up the transport and network layers. Some candidates also labelled the layers in the reverse order which resulted in them obtaining fewer marks. Part (a) (ii) was poorly done. Approximately 5 per cent of the candidates were able to correctly describe the purpose of the three layers.

In Part (b), only a few candidates were able to accurately define a page fault and describe how it was handled by the operating system. Most candidates described the paging process and did not answer the question. Many of the responses incorrectly stated that a page fault occurs when there is an error on the page. The correct answer is that *a page fault occurs when a program tries to access a page that is not currently mapped to RAM.*

In Part (c), candidates generally confused GPRS with GPS and consequently answered the question incorrectly. GPRS is a technology used to connect wirelessly to the Internet and can be used by point-of-sale devices or other devices needing wireless connectivity to the Internet. Part (d) was fairly well-attempted. However, many candidates did not clearly describe the steps involved when a process moves from the running state to the ready state.

Question 6

This question tested candidates understanding of various networking and operating system concepts. It was attempted by most candidates. Responses were fair with average scores between 11 and 15 marks. A few candidates got 21 – 25 marks.

Parts (a) to (g) and (k) (iii) were well done by most candidates. Parts (h), (i), (j), k (i) and k (ii) were not answered well by most candidates. They were unable to provide proper explanations which indicate the need for greater depth of knowledge in these areas, for example, client-server versus peer-to-peer configurations.

Based on the responses obtained for this question, it is clear that candidates need to become more familiar with the technical terms used in the syllabus.

GENERAL COMMENTS

Paper 03 - Internal Assessment

In general, performance on the School-Based Assessment was very good. The performance on Unit 1 was roughly the same as in 2009. However, the performance on Unit 2 increased by more than 10 per cent compared with 2009.

DETAILED COMMENTS

UNIT 1

Some candidates wrote their programs using separate program files which were saved and printed individually rather than using separate functions within the same program. There were candidates who did not provide printed evidence for the headings which were included in their submissions, for example, a heading stating 'PSEUDOCODE' often had no printout showing an algorithm in pseudocode.

Some candidates included trace tables which were not required. Teachers need to pay careful attention to the requirements of the syllabus. Other candidates used data flow diagrams instead of narratives for describing algorithms. There were also candidates who utilised one main function and no function decomposition was seen in some samples.

Some candidates submitted a pseudocode which was identical to their programs. A pseudocode should be produced before the program is written and will generally look different from the source code of the program. In some samples, pseudocode algorithms were not written using correct logic, for example, variables controlling iterations were not incremented and *while* statements were not closed with *Endwhile*.

There were candidates who submitted the same pseudocode, program code and test data. Subheadings were often missing in some of the documents submitted and the word limit was exceeded in some instances. Flow charts in many samples made use of system symbols rather than the normal flowchart symbols and some test plans did not use erroneous and extreme data for testing. Screen images (screen shots) to illustrate the program output were not shown in many samples. These are important for verifying the output of the program.

The problem statements in some samples were either very trivial or too detailed. Candidates are reminded that the source code for programs must be printed from the C development environment and that a narrative must be given which is related to the problem solution.

Unit 1 samples often included entity-relationship diagrams and data flow diagrams; however, these are only relevant to Unit 2.

UNIT 2

Some candidates were unable to differentiate between functional and non-functional requirements. Functional requirements are features the system must provide. Non-functional requirements are constraints that apply to the system, for example, it must perform a transaction in less than two seconds.

Attributes of entities were not included in the entity-relationship diagram by most candidates the entity-relationship diagrams and data flow diagrams were often inconsistent with the problem definition.

It should be noted that user technical manuals are not required and that background information and abstract information should be kept to a minimum.

Some problem definitions were often too wordy, and, in some instances, made no sense in the context of the system developed.

Candidates are reminded that all design diagrams must be drawn electronically, for example, system structuring and user interface design. Programs must be written in C code and printed output must be done from the C development environment.

When preparing their reports, candidates should use subheadings which are specified in the syllabus. The ordering of the headings should also be followed.

RECOMMENDATIONS

Candidates are encouraged, as part of their examination technique, to read questions carefully before answering, and to respond with sufficient detail that is commensurate with the marks indicated for each question. Candidates need to answer questions in the context given instead of simply regurgitating notes. Past-paper questions should be carefully studied to understand how syllabus items can be tested.

Teachers need to prepare students adequately for the three modules of each unit. Algorithm Development, Computer Programming, and Data Structures account for three of the six modules of the Computer Science syllabus so teachers need to cover these modules in more detail, scheduling as many practical sessions as possible. This can take the form of

- classroom discussions geared to solving specific problems or developing algorithms to solve given problems
- laboratory exercises involving the writing and testing of computer programs and experimentation with ‘buggy’ programs
- using the Internet to download tutorials on programming and sample code to experiment with.

The poor performance by candidates on the Computer Architecture and Organization module can be handled by adequate coverage of the syllabus items. This module does not involve computer programming. Candidates need more practice with numerical problems such as those involving decimal to binary conversion and vice versa, as well as problems involving one’s and two’s complement.

With respect to the School-Based Assessment, teachers need to provide better feedback to studentss. The School-Based Assessment should be viewed as a process with well-defined milestones. As students submit work for these milestones, teachers need to provide adequate guidance, correcting errors made and indicating clearly what is required for the next milestone.