C A R I B B E A N E X A M I N A T I O N S C O U N C I L

REPORT ON CANDIDATES' WORK IN THE CARIBBEAN ADVANCED PROFICIENCY EXAMINATION®
MAY/JUNE 2014

COMPUTER SCIENCE

GENERAL COMMENTS

This was the sixth year in which the revised syllabus was examined. 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 School-Based Assessment (SBA), was examined by teachers and moderated by CXC.

In each unit, Paper 01 consisted of multiple-choice questions which 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.

Approximately 93 per cent of candidates obtained Grades I─V in Unit 1 and approximately 92 per cent of candidates in Unit 2 obtained Grades I─V. Overall, there is still need for improvement in the quality of responses for the programming questions in both units.

DETAILED COMMENTS

UNIT 1 – FUNDAMENTALS OF COMPUTER SCIENCE

Paper 01 – Multiple Choice

Performance on the 45 multiple-choice items on this paper produced a mean of approximately 61.20 out of 90, standard deviation of 14.64 and scores ranging from 24 to 90.

Paper 02 – Essay Questions

Section A – Computer Architecture and Organization

Question 1

Part (a) tested candidates' ability to draw a clearly labelled block diagram of a 2-to-1 line multiplexer. Most candidates attempted this question gaining at least three of the five marks. Many candidates lost marks mostly through failing to realize that a 2-to-1 line multiplexer has one select line. Candidates also lost marks via omission of the select line and improper labelling.

Part (b) required candidates to define the term *logic gate*. Most candidates wrote that logic gates take inputs and provide outputs according to their logical rules. Candidates did, however, lose marks by failing to note that logic gates are elementary building blocks/components of digital circuits.

Part (c) provided a circuit diagram and asked candidates to provide the truth table associated with the diagram. This part was fairly well done. Candidates are however encouraged to maintain order when displaying the input values for the truth table (for example, 0 0, 0 1, 1 0, 1 1).

Part (d) required candidates to indicate which of *decoder, flip-flop and multiplexor* are associated with a single input. Most candidates were able to identify at least one out of the two correct devices (*flip flop and multiplexor*). Candidates were not required to provide definitions for the structures identified.

For Part (e) (i), candidates were required to provide the number of digits that a 3-bit decoder would display. Many candidates lost marks by answering 6 (2 x 3). A decoder can provide 2" outputs where n is the amount of different inputs. Therefore the number of outputs would be 8.

For Part (e) (ii), candidates were required to examine the digital figure provided and list the segment letters that must be switched on to display the number 5. This part was well done. Candidates provided most or all of the segments needed. Some candidates provided the segments that should be off and were awarded no marks for their response.

For Part (f), the responses provided by candidates were mostly correct although some candidates included sign and magnitude alongside the complement process and added a sign bit before inverting the value. Candidates must be mindful of the number of bits required for the question as those whose final responses were not represented using 4 bits lost marks as a result.

Part (g) asked candidates to determine what decimal number was being represented by the given binary string. Many candidates did not seem to have knowledge of this process and some candidates simply treated the binary as an absolute value and converted it to its decimal equivalent. Candidates were expected to have displayed the component parts of the string showing *0 for the sign, 011 = 3 for the exponent* and *10110 for the mantissa*. Many candidates failed to recognize that the mantissa is a fraction and should be written as 0.10110.

Question 2

Part (a) tested candidates' ability to describe in the correct order, the three main activities in an instruction cycle. Candidates were able to correctly identify- in the correct order- the three main activities that take place in an instruction cycle as fetch, decode and execute. Many candidates attempted to discuss the activities outlined but gave vague explanations of each.

In Part (b) (i), candidates were required to give one similarity and one difference between RAM and hard disks. This part was well done.

Part (b) (ii) tested candidates' knowledge of registers used in the CPU of a computer. Few candidates were able to provide the correct responses to this part. Most neglected to thoroughly explain the reasons for including registers in the CPU.

For Part (c), almost all candidates attempted to describe situations where a Supercomputer, PDA and Mainframe can be used. However, many candidates gave definitions or explanations of these categories of computers rather than their uses.

Part (d) was very well done.

For Part (e), whilst the majority of candidates correctly identified port connectivity, speed and software incompatibility as reasons why current computers may not be able to work efficiently with new storage devices, very few adequately discussed these points resulting in their inability to obtain full marks.

Section B – Problem Solving with Computers

Question 3

This question tested candidates' ability to:

- Identify and give examples of the main control constructs used in programming.
- Generate an algorithm to select the favourite subjects of 100 students.
- Generate an algorithm to calculate the sum of the multiples of six and seven between 100 and 250.

For Part (a), the majority of candidates described the control constructs and gave an example of each. This part was fairly well done.

Part (b) was done well by the majority of candidates. A few candidates did not initialize the variables needed to perform the counting operation. Some candidates also used the selection construct improperly.

Part (c) proved challenging for some candidates. Those who did not complete the question had difficulty testing correctly for the multiples of the given numbers.

More attention should be focused on the following:

- correctly using the sequence, selection and iteration control constructs
- using functions such as modulus to test numbers for specific properties
- correctly selecting the limits of loops based on given scenarios

Question 4

Part (a) tested candidates' ability to write an algorithm using the iterative control construct to solve a given problem; Part (b) required candidates to represent the algorithm with the use of a flowchart and Part (c) tested candidates' ability to trace an algorithm to determine its output.

More than 90 per cent of the candidates attempted this question and gave satisfactory responses.

Part (a) was fairly well done. The weaker candidates confused the data in the example given as a solution to their algorithm. Some candidates used the iterative control construct improperly, using the "$" in the IF statement as well as failing to initialize the variable for the running total. Some candidates also answered the question using the C programming language rather than the construction of the algorithm.

Part (b) was attempted by most candidates and was generally well done. Some of the weaker candidates used the flowchart symbols incorrectly and some excluded the iterative statement in the flowchart.

Part (c) was also attempted by the majority of candidates and they performed very well. In most cases, they scored full marks. Some candidates did not change the value of the variable 'x' for the trace table which resulted in incorrect values.

Section C – Programming

Question 5

This question tested candidates' knowledge of the stages in the translation process of a program, and the purpose of the watch window in the C compiler. Candidates were also required to write a C function based on a given scenario and apply knowledge of FILE processing.

Part (a) was poorly done by most candidates. Candidates attempting this part seemed to be familiar with the lexical analysis stage only.

Part (b) was poorly done. Most candidates simply did not know the purpose of the watch window in the C compiler.

Part (c) was poorly done. Candidates seemed not to know the difference between a *main* function and a *programmer defined* function, hence many wrote their solution using main. Those candidates who did well were able to write a proper function header, sum the elements of an array and return its total. Candidates lost marks if they did not declare and initialize their variables.

Part (d) was poorly answered. Most candidates who attempted this part of the question knew how to create a FILE pointer. However, candidates had difficulty writing the appropriate structure of *fscanf*, *fprintf*, and *fopen* statements. Candidates seemed not to know when to use the different modes of opening a file, for example, (r, w, a, r+, w+, a+).

Question 6

This question was designed to test candidates' ability to:

- Explain the importance of good programming style.
- Interpret source code and utilize their knowledge to formulate the output of given code.
- Write a C program that tests their knowledge of program structure, variable declarations, input and output of variables, and their ability to formulate an equation to solve the addition of taxes on an existing price.
- Write C code that tests their ability to read and print numerical values after performing an arithmetic calculation to solve the average of those numbers.

Part (a) was generally well done. Some candidates failed to explain the importance of indentation clearly and lost marks. A few candidates confused the purpose of indentation and associated it with a benefit towards the computer or compiler.

Part (b) was well answered; only a few candidates did not elaborate their answers properly.

Part (c) was not clearly understood by many candidates; the majority failed to provide the actual output that was being requested. The majority of candidates took the approach of providing a trace table and some simply attempted to show some sort of calculation to arrive at their answers.

Most candidates failed to analyse the loops properly. Upon entering the *while* loop, candidates performed well in obtaining the result of the first stage but failed to re-enter the *for* loop with the appropriate data hence, obtaining wrong data thereafter.

Part (d) was generally well understood by most candidates but some failed to write correct syntax either in the form of program structure: missing out # or braces; variable declarations: declaring as *int*; reading of inputs: missing out the *&*; and in their output: not formatting their result as *%6.2f*. Some candidates failed to analyse the mathematical problem as most failed to divide the *tax* by 100.

Part (e) seemed to be unclear to most candidates as it pertains to the required result. Most candidates read variables properly and solved the result but failed to provide their answer as a valid average of type *float*. Some candidates failed to apply proper precedence of operations by omitting brackets and some even ended up dividing the sum of all numbers read by values such as 2 and 3 when the question clearly asked for 4 numbers. The majority of candidates failed to provide their result as a *float* data type.

Candidates are encouraged to pay close attention to the use of proper coding syntax when asked to write C programming code. Candidates are also encouraged to carefully read and analyse the questions to fully understand what is being requested of them.

Paper 03 – School-Based Assessment (SBA)

General Requirements

Students were expected to choose a problem for which a software solution was appropriate; develop and present the solution in a logical way using correct grammar and appropriate jargon at all times by doing the following:

- Create an algorithm for the solution using modules, sequencing, selection, assignments, and iteration (bounded and unbounded).
- Represent their algorithms using narrative format and also as either a flow chart or pseudo-code.
- Implement the algorithm in C using various data structures such as arrays, struct, strings and files, with no less than five functions as independent units.
- Ensure that the source code produced, matches the algorithm.
- Create a test plan with exhaustive data set, test and produce test results and appropriate error messages.

The following specific aspects of the project were assessed:

- Definition of problem;
- Narrative and flow charts or pseudocode;
- Coding of program;
- Testing and presentation;
- Communication of information.

For each component, the aim was to find the level of achievement reached by the student. It is recommended that teachers make the assessment criteria available to students at all times.

Definition of problem

Many students were able to accurately describe the problem complete with the description of the current system and giving background to the problem, the issue or problem, recommended solution and examples of what takes place when the problem arises. However, some students' projects only included partial definition of activities of the proposed system that they intended to create without completely describing the current system and the problems which are to be solved using software. In addition, some problems that were identified could not be solved by using a computer program. For example, filling out of a form is a time-consuming and error-prone process for both paper-based and computer-based systems. However, there are advantages of using computer-based forms that were not identified.

Narrative and flow charts or pseudocode

Narratives were fairly well written in most samples. However some samples:

- Provided only a partially correct description of the algorithm as a solution.
- Provided descriptions of an intended module or unit instead of describing the step by step procedure that would solve the problem as required by an algorithm.

Pseudocode algorithms were generally well done. Samples which included flowcharts in some instances incorrectly used some symbols. For example, system flowchart symbols were used. In others cases, diagrams were poorly presented, had symbols with no labels (such as yes/no for the decision symbol), had no connector symbols to connect diagrams to the next page. Charts were therefore difficult to follow. Most students produced samples using pseudo-code algorithms. Some samples were clearly modified copies of the programming source code, and a few almost identical copies of the source code. It is therefore recommended that in the future, students be encouraged to develop their algorithms independent of the programming code.

Coding of program

Students' projects demonstrated that they were comfortable with procedural C programming language. This was quite evident by the responses of the various projects. Most programs were logically written and properly decomposed. Nevertheless, some students:

- Did not use functions as independent units in their programs and yet some teachers still awarded full marks.
- Used too few data structures in their programs.
- Did not demonstrate appropriate use of the concept of structured programming.

Some students did not print their programs from the C compiler, instead choosing to make print screen copies of their projects. Students are therefore reminded to print their source code directly from the C compiler as transferring the code to a word processor often changes the code. For example, transferring to a word processor may cause adjustments to the spacing and cause strange symbols to occur in the code, making it difficult to read. This may also lead to students losing marks for inappropriate programming style and documentation. In addition, marks were lost because some students did not include adequate comments at key areas in their source code, and they used a poor indentation style that did not make the code easier to read. However, this section was generally done well by most students.

Evidence that code matches algorithm

Most students were able to obtain full marks on this section as this was well done.

Evidence of file manipulation

Students were required to provide code that included a file and show evidence of file manipulation that is, open file; write to file; read from file; append file and close file.

This section was well done. However, some students were awarded marks by the teacher even though there was no evidence of file manipulation and in some cases the use of files was non-existent or was declared but not used, and the teacher still awarded full marks. Projects that showed screenshots of the file data being added, edited and deleted were awarded full marks.

Testing and presentation

Students were required to prepare a test plan with an exhaustive data set (test data) to test their programs. Students were also expected to use the test data set to produce test results with normal input giving correct results, extreme input giving correct results and erroneous (abnormal) input giving appropriate error messages. The majority of samples moderated this year did not have a suitable range of test data. However, a few students:

- Did have a test plan and screen shots while others provided screen shots without any test plan.
- Did not include all testing criteria (normal, extreme and erroneous) in their test plan.
- Had test results but did not have a clear test plan.
- Had no test results but were awarded marks by teacher.

- Showed test results but did not include actual screen shots of the working program. Testing was not done using the test data outlined in the test plan.
- Tested their menus with the appropriate test data, but all input must be tested in the same way.

Generally, this year's SBA projects were well presented.

Recommendation

Teachers need to ensure that students use the headings as outlined in the syllabus. In addition, students should follow the order in which these heading occur in the Form CSCI 1-3. They should also ensure that students check that the numbering in the table of contents corresponds to the numbering in the document and that students provide information in a logical way using correct grammar and appropriate jargon at all times in the presentation of their projects.

UNIT 2 – FURTHER TOPICS IN COMPUTER SCIENCE

Paper 01 – Multiple Choice

Performance on the 45 multiple-choice items on this paper produced a mean of approximately 62.19 out of 90, standard deviation of 13.09 and scores ranging from 17 to 86.

Paper 02 – Essay Questions

Section A – Data Structures

Question 1

Part (a) tested candidates' knowledge of the creation and deletion of a *STACK* (*ADT*). The majority of candidates was able to correctly identify that a stack only needed to exist in order to execute the *Destroy Stack* function. The modal mark was two in this section.

For Part (b) (i), candidates were expected to perform the basic *PUSH* and *POP* operations on a stack; (Module 1: Specific Objective 3). This question tested analysis and interpretation of the given operations. Generally, candidates were able to acquire total marks when they clearly illustrated the contents of the stack at various stages as the operations progressed. In a few cases candidates focused on the computation and neglected to show the contents of the stack. Marks were not awarded in those cases.

The responses to Part (b) (ii) were generally accurate. Most candidates were able to identify that there should be at least two elements in the stack in order for the *MULT* and *ADD* operations to work effectively. It should be noted that there were cases where candidates gave a vague response, for example, 'Stack should not be empty.' Marks were not awarded in these cases.

For Part (b) (iii), candidates were expected to analyse the above scenario and deduce that the contents of the stack would be reduced to one element and eventually the *ADD* operation will attempt to *POP* from an empty stack.

For Part (c) (i), the majority of candidates obtained four out of five marks because they were able to accurately identify that the *ENQUEUE* operation adds an element to the rear/tail of a queue and the *DEQUEUE* operation removes/retrieves an element from the front of the queue.

Candidates performed poorly on Part (c) (ii). The majority of their algorithms were incomplete and incoherent. The *DEQUEUE* function was not implemented properly and the iterative construct and its condition were in most cases incorrect or omitted. Generally, candidates were expected to:

- Initialize a 'count' variable
- Implement an iterative construct which will terminate when the queue is empty ('Dequeue' the queue correctly and Increment count)
- Return/Display count

Question 2

Part (a) tested candidates' ability to explain how two numbers, 5 and 15, would be inserted into a singly linked list and then to represent the final list in the form of a diagram. Most candidates were able to give some level of explanation, however many were somewhat vague. In most cases they demonstrated the concept of a link between nodes but often were not clear on the sequence of steps involved in inserting items into the linked list. Candidates needed to clearly indicate in their explanation that a head was present and linked to either a node or null. Additionally, candidates needed to indicate that a new node must be created first then the number is stored in it and not merely "insert the number into the list".

About 80 per cent of the candidates were able to produce reasonable diagrams to represent the nodes, however many did not correctly represent key components. For example, some candidates omitted arrow heads, while others had the head pointer arrow in the wrong direction. In some cases, the head pointer was drawn to look like a node; candidates needed to demonstrate some distinction between them. In other cases the notation used for the node was incorrect in that it did not clearly show the two parts, that is, the data and the address fields.

Part (b) tested candidates' ability to declare an integer array, read values into it and then search for the presence of a particular value.

The majority of candidates were able to correctly declare an array. Candidates almost always indicated the correct data type as well as the array size. However, some candidates used the wrong name for the array although a specific one was given in the question.

In terms of storing items in the array which was the second requirement of this part of the question, most candidates demonstrated the need for a loop in order to perform this task efficiently. They were also able to produce the correct or near correct code to accept values and assign them to the array locations. However, many candidates omitted one or two key components such as the ampersand or the index. In some cases they also used the wrong format specifier instead of *%d*.

In the final section of Part (b), most candidates were able to produce C code to accept the search key. Some of them were able to write reasonable code to conduct the search, however quite a few used the wrong upper limit for the loop. They could have used < 100 since the C programming language starts to index an array from 0 causing index value 99 to represent the 100^{th} location.

For the comparison many used incorrect notation for 'equal to'. Candidates should note that two equal signs '==' are used for this purpose while a single '=' is used for assignment.

For the last statement which should state 'Key Not Found', many candidates placed it within the loop so that the error message would be printed repeatedly. Other candidates did not have a statement to allow the program to break out of the loop after the key was found. This meant that the loop would continue processing unnecessarily after the key was found.

Section B – Software Engineering

Question 3

In general, almost all candidates attempted this question.

For Part (a), many candidates incorrectly wrote that a deliverable was the end product of the Systems Development Life Cycle (SDLC) to be 'delivered' to the customers. The majority of candidates did however identify a deliverable as some form of documentation or tangible/intangible result of each phase of the SDLC. Candidates lost marks for not indicating that a deliverable needed to be signed off and used as input for subsequent phases.

In Part (b), candidates responded well to the reasons why an information system may need to be changed. Candidates easily identified issues that would cause a system to change including change in requirements, inefficiency and outdated systems. Yet candidates could have done better by suggesting how a new system with modern updated technology would alleviate these issues.

In Part (c), while some candidates were able to correctly identify the factors: operational, technical, economic and legal, in some cases candidates' responses lacked proper explanation of the terms and often candidates would confuse operational feasibility with technical feasibility.

For Part (d), the ERD was done quite well by most candidates. It should be noted that many candidates who should have received 12 marks lost marks for simple errors including:

- Plural entities (ending with 's')
- Not indicating the primary key by underlining the attribute
- Incorrect cardinalities including the notation

Question 4

This question generally tested candidates' knowledge of the design of input forms and of data flow diagrams (DFDs). Part (a) (i) was generally well done. However, some candidates provided one example only of data in the required formats.

Part (a) (ii) asked candidates to justify their selection of one of four input options. However, many candidates did not match the input option to a specific data item. Others misinterpreted the question and recommended different types of user interfaces for instance command-line or menu based.

Part (a) (iii) was poorly done. Most candidates incorrectly described range verification checks and did not relate this part of the question to the previous parts.

Part (b) tested candidates' ability to create a data flow diagram based on a given scenario. Most candidates provided satisfactory responses, however many candidates incorrectly labelled the processes as departments; drew context diagrams which included each department listed as a source/external entity; and included departments along with their processes as separate symbols.

Candidates also lost marks for the data flow in the diagram because they incorrectly used verbs to describe the flow of data. Additionally, marks were not awarded when candidates used incorrect symbols for the four data flow diagramming objects (source/sink, data store, data flow, process).

Section C – Operating Systems and Computer Networks

Question 5

This question tested candidates' knowledge of network devices, including: router, switch and modem along with their roles; hybrid network topologies, fibre optic cables, the IEEE802.11b network, VOIP, and GPRS.

Part (a) (i) dealt with setting up a small network using the following network devices: modem, router, and switch. Most candidates attempted this question, however, even though the devices were shown, some of them did not connect the devices correctly and critical ports were not shown.

Part (a) (ii) tested candidates' knowledge of the role of the modem, switch and router. Most

candidates demonstrated an understanding of the role of the modem being analog to digital conversion and vice versa. However, many candidates failed to adequately state the role of the router and switch.

Part (b) tested candidates' knowledge of the hybrid network topology. This question was well done. Almost all candidates were able to provide a correct definition of a hybrid network topology.

Part (c) tested candidates' knowledge of the advantages and disadvantages of fibre optic cables. This part was poorly done since candidates failed to adequately describe the advantages and disadvantages.

Part (d) was poorly done since many candidates failed to clearly explain how data is communicated in an IEEE802.11b network. Many of them did not demonstrate understanding of the use of wireless access points in this kind of network.

Part (e) required candidates to explain why the quality of voice over IP (VOIP) communication might be different from telephone communication service as offered by a telephone company. Most candidates demonstrated understanding that VOIP uses the Internet but failed to discuss the deterioration of the quality of the call where there are Internet problems or data packet delays.

Part (f) tested candidates' knowledge of the main purpose of GPRS. This question was fairly well done. Many of the candidates were able to link the term *GPRS* with the transmission of internet packets through a cellular network.

Question 6

This question tested candidates' knowledge of operating systems, security of data and device management. In addition it tested their knowledge of

- process management and in particular, process states
- interrupts and their application to scenarios
- scheduling processes with emphasis on pre-emptive scheduling.

For Part (a), most candidates were able to correctly identify that batch systems compiled jobs in groups. However, they were unable to specify that these jobs would be processed at a later time. The majority of candidates was also able to say that multi-user systems allow the use of many users at the same time but failed to identify their processing power.

For Part (b), most candidates recognized that a process may be blocked but were unable to define *a blocked process as a process that is paused, pending an event*. Candidates also failed to specify that *a running process is one where instructions were being executed by the CPU*. This part of the question was poorly answered by most candidates. Many candidates understood that the game would be halted to process the interrupt but most candidates did not identify the type of interrupt. Candidates also failed to recognize that a context switch occurs, that is, saving the state of the running process in registers for later resumption, and that the game process resumes after the processing of the interrupt.

Part (c) was poorly answered. While some candidates recognized that pre-emptive scheduling meant that the scheduler had the power to pre-empt the tasks, they did not specify that the processor would resume the previously running tasks.

In Part (d), the majority of candidates was able to correctly identify three ways of protecting data. However, some candidates were not able to sufficiently discuss their methods in order to gain full marks.

In Part (d), candidates were able to identify the type of software being used as device drivers. However, most candidates did not give the definition of a device driver as *the set of instructions used*

to communicate with the device. They also neglected to mention that each device driver uses the same protocol to communicate with the OS.

Paper 03 – School-Based Assessment (SBA)

Requirements

Each student was expected to choose a problem for which a software solution exists and then develop the software using software engineering techniques. In particular, the student was expected to demonstrate appropriate choice of the tools and techniques used in the analysis of the software to be developed. They were then expected to design, code, and test their software using appropriate techniques.

General Comments

- Students should follow the order laid out in the criteria for marking when arranging the sections of the SBAs.
- Teachers should avoid using red ink pens to correct SBAs.
- Teachers should ensure that each SBA is clearly labelled with the student's name and centre number.

Marking Criteria

Specification of Requirements

Definition of Problem

Students were required to give a complete and accurate description of the problem. Most students handled this section fairly well but there were a few students still unclear as to what to include in their definition of the problem. A brief description of the context in which the problem occurred is required but details about actual problems staff/clients face and proposed steps to correct such problems must be emphasized.

Techniques of Analysis

Students were required to identify techniques of data collection and analyse the data collected.

- Most students were able to state the various techniques of data collection; however, some students were not able to justify their selections. Several students misinterpreted justification to mean definition. This is incorrect. Students should clearly explain why the chosen technique was used as it relates to the business/company/institution and not regurgitate advantages of the technique.
- Proof of analysis should also be given. All proof must be included directly after the analysis not in the appendix. For example, sample questions from the questionnaires and/or interviews should be included. It is understood that it may be difficult to include proof of observation; hence, if this technique is used, students should clearly describe what was observed.

Use of Data Flow Diagrams and E-R Diagrams

This section of the SBA was poorly done by most students.

Context Level Diagram

Students were required to give a complete and accurate diagram of all relevant entities and data flows.

- In some instances, incorrect symbols were used for entities and processes.
- Some data flows were not labelled.
- Data stores/files were incorrectly included in the Context Level Diagram.

Level 1 Diagram

Students were required to give a complete and accurate diagram with all relevant processes, data flows and major data stores.

- Most students are unaware that the Level 1 diagram is an expansion of the context level diagram. Hence, new external entities were created for the Level 1 diagram, as well as new data flows were created for previously used entities.
- Students had links between
  - data stores and entities
  - data stores and data stores
  - entities and entities

ER Diagram

Students were required to give a complete and accurate diagram of all relevant entities and relationships.

- Most students did not use the correct symbol for a relationship (a diamond).
- Some students did not include attributes for the entities.

Functional and Non-Functional Requirements

Students were required to give a complete and accurate description of all requirements.

- For the functional requirements, students did not clearly state what the system is expected to do; instead, they stated what the user will be doing. An example of a good functional requirement is *the system will be able to delete a patient's record.*
- For the non-functional requirements, students did not state the limitations of the system. An example of a good non-functional requirement is *the system can only store 1000 patient records*.

Design Specification

Students were required to give a complete and accurate system structuring diagram containing all processes and a description of the user interface, report design, algorithm design and appropriate data structures. Most students handled this section fairly well. However, the following should be noted:

- Students should not just include screens shots of interface and report design but also a justification in order to gain maximum marks.
- Narratives will not be accepted as an algorithm. Students are reminded to submit pseudocodes.

Coding and Testing

Students were required to produce a complete and accurate C program solution for the problem stated in the Definition of Problem section. Most students produced programs that achieved good functionality. Students should be aware that:

- All code must be written in procedural C. No other programming language will be accepted. Code must also be printed from the compiler, NOT transferred to a word processor before printing.

- Soft copies will not be marked.
- Most students did not include enough screen shots (and in some cases none at all) to support their functioning of the code.
- Some codes presented did not match the screen shots given. In such cases students were not awarded any marks.
- Test plans should be written in a tabular format and it should include normal, extreme, erroneous and incompatible data.
- All input data must be tested.
- Test results without related test plans will not be awarded any marks.

Communication and Presentation

This section of the SBA is often overlooked by students. Teachers and students are urged to pay close attention to their use of grammar and the overall presentation of the SBA.

Recommendation

Teachers should ensure that students are fully prepared for the examinations in both units. Poor performance in some modules of the syllabus indicates that more time needs to be allocated to these areas.