CARIBBEAN EXAMINATIONS COUNCIL

REPORT ON CANDIDATES' WORK IN THE ADVANCED PROFICIENCY EXAMINATION

MAY/JUNE 2011

COMPUTER SCIENCE

Copyright © 2011 Caribbean Examinations Council
St Michael Barbados
All rights reserved

GENERAL COMMENTS

This was the third 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 45 multiple-choice questions that were designed to test candidates' breadth of understanding of the syllabus. Paper 02 consisted of six 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, 87 per cent of the candidates obtained Grades I to V, a 12 per cent improvement over 2010. In Unit 2, 87 per cent of the candidates obtained Grades I to V, a five per cent decline compared with 2010. In general, performance on the SBA was very good. In Unit 1, there was a slight decline in the performance on the SBA compared to 2010. The average mark was 68 per cent compared to 70 per cent in 2010. However, there was a significant decline in the performance on the School Based Assessment in Unit 2 compared with 2010. The average mark was 57 per cent compared with 66 percent in 2010. Nevertheless, in both units, the average mark was an improvement on the corresponding 2009 average.

Performance on questions involving computing programming continues to be poor. However, there was a reasonable improvement in scores for most of the programming questions in Unit 1 compared with 2010. Performance on the programming questions in Unit 2 was roughly the same as in 2010. Despite the improvements noted, it is clear that many candidates still need to get more experience 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 – Fundamentals of Computer Science

Paper 01- Multiple Choice Questions

Performance on the 45 multiple-choice items on this paper produced a mean of 54 out of 90 with scores ranging from 18 to 86.

Paper 02 – Essay Questions

Section A – Computer Architecture and Organization

Question 1

This question examined candidates' knowledge of truth tables, logic gates, circuit diagrams, flip-flops, multiplexors, decoders, and the internal representation of data on a computer. It also required candidates to apply their understanding of a decoder to solve a simple problem.

Overall performance on this question was poor and the majority of candidates obtained less than 10 marks.

In Part (a), only a few candidates were able to correctly convert the truth table to a circuit diagram. Some candidates even used secondary logic gates (NOR, NAND) in their responses which were unnecessary. In Part (b), most candidates performed well and got full marks or at least half of the marks allocated for this question.

In Part (c), many candidates stated that a flip-flop can be used to store one bit of data. However, only a few of them went on to fully explain what is a flip-flop and hence did not obtain full marks. Many candidates obtained full marks for their explanation of the multiplexor.

Part (d) (i) was generally well done. Most candidates gave a correct diagram for the decoder; however, some candidates did not label the diagram and a few attempted to draw a circuit diagram. The performance on Part (d) (ii) was fair. A few candidates drew circuit diagrams instead of using the block diagram of the decoder. Most candidates did not explicitly state that there are two inputs and that the bulbs have to be connected to the outputs. Also, some candidates did not explain how the two inputs were connected to the four outputs (the bulbs).

Part (e) was fairly well done. The majority of candidates were able to convert the decimal value to binary, but a number of them had difficulty finding the two's complement. Part (f) was poorly done. Many candidates simply converted the binary representation to its decimal value. Some of them correctly converted the 3-bit exponent to decimal but were unable to use this value to correctly complete the

calculation. A few candidates assumed that the decimal point was to the right (00101.) rather than the left (.00101).

There is cause for concern with candidates' performance on this question, particularly on Parts (e) and (f) since the representation of data on a computer is one of the most basic and straightforward items in the syllabus.

Question 2

This question tested candidates' knowledge of registers, Central Processing Unit components, storage devices, cache memory, instruction formats and addressing modes.

The question was answered fairly well and there was an improvement in responses compared with to those given for a similar question in 2010.

In Part (a), candidates' were generally weak in their understanding of registers found in the CPU. Candidates were expected to mention registers such as the MAR, MBR, PC, AC and IR.

The responses to Part (b) were generally fair. However, many candidates only compared main memory and the two given storage devices in terms of capacity or access speed alone. They thus lost half of the marks for this question.

Part (c) was generally well done since this is a topic well understood by most candidates. The Arithmetic and Logic Unit (ALU) and Control Unit (CU) were popular responses for this question. However, the function of the components stated (e.g. the CU) was not properly stated in many responses.

The responses to Part (d) were generally fair. However, many candidates could not clearly explain how cache memory increases the efficiency of data retrieval (by keeping frequently accessed portions of main memory in the cache).

The responses to Part (e) were generally poor. Candidates need to spend more time understanding the various types of addressing modes specified in the syllabus.

Section B – Problem Solving with Computers

Question 3

This question tested candidates' knowledge of the stages in the problem-solving process and their skills in constructing a flow chart from an algorithm as well as tracing through and identifying errors in an algorithm. The question also examined candidates' ability to understand and modify an algorithm based on given specifications.

Overall, the question was well done and it appeared that many candidates understood what was required. The mean performance on this question was the best of all the questions in Paper 02.

In Part (a), most candidates were able to identify three stages in the problem-solving process. The stages were sometimes incorrectly given as the stages of the systems development life cycle. Candidates were often unable to clearly describe each stage, particularly the analysis stage (which seeks to determine how the problem manifests itself).

Most of the candidates gave good responses for Part (b). However, some of them drew incorrect symbols to represent input/output and processing steps and some were uncertain about how to represent a loop in the flow chart. In a few cases, programming code was included with the symbols. Candidates need to spend more time constructing flow charts and should make sure that they understand the purpose of each symbol used in a flow chart.

In Part (c), candidates were able to correctly trace through and identify the lines of the algorithm containing the errors. However, many were unable to correct the errors. It is recommended that candidates trace through an algorithm after modifying it to ensure its correctness.

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 properties of a well-designed algorithm.

In Part (a), many candidates were unable to identify three properties of a well-designed algorithm. This was rather unfortunate since it involved straightforward recall of a syllabus item.

In Part (b), many candidates were unable to trace through the while loop correctly and generate the triangular shape of asterisks. In addition, there was some confusion between the use of *print* and *println*. However, the question clearly stated that *println* did the same thing as *print* except that subsequent output started on a new line. Only a few candidates were able to correctly calculate the values for *y* on each repetition of the *while* loop. This indicates a weakness in basic arithmetic skills. Candidates need to spend more time tracing through algorithms since this is a fundamental skill in computer science.

In Part (c), some candidates were unable to write correct algorithms in terms of looping constructs, accumulating the sum, and finding multiples of 11. It was clear from their responses that many candidates were unaware that a *while* loop can be bounded or had a misconception that a *while* loop is always unbounded. A loop is bounded if the number of iterations is known beforehand or can be calculated beforehand. It is sometimes possible to calculate the number of iterations of a *while* loop before it is executed; thus, in these situations, the *while* loop is bounded.

Section C – Programming

Question 5

This question tested candidates' knowledge of the stages of the translation process and of the advantages of using a modular approach in programming. Candidates were required to write a C function which took two parameters, one of which was an integer array. The question also tested candidates' ability to take a specification of a program and write a C program which achieves the desired functionality. The program required candidates to know how to read and write from text files.

Part (a) was answered fairly well by candidates. Most of them were able to list the stages of the translation process.

Part (b) was fairly done. However, a few candidates confused the term *modular* with the *modulus* operation. Also, several candidates were vague or overly general in their responses.

Part (c) was poorly answered. Many candidates gave a program in their response rather than a function and lost marks because of this. It is fairly straightforward to find the sum of the numbers in an integer array and the code for this functionality should have been placed in the body of the function. In addition, candidates were expected to specify the parameter list and return type of the function as well as return the sum of the values after it was calculated inside the function.

Part (d) was also poorly answered. Candidates seemed to lack skills of analysis to understand the functionality that the program was required to provide. For those who were able to write a reasonable program, it was clear that most of them were unable to manipulate text files. Many of the candidates did not adhere to the syntax of the C language, particularly in the use of semicolons, ampersands, format specifiers and appropriate quotation marks. Candidates need to spend more time writing programs from a given specification using the programming language features outlined in the syllabus.

Question 6

This question tested candidates' understanding of good programming style such as the use of white space and proper indentation. It also tested their understanding of the term *debugging*. Candidates were required to write a program to determine what type of triangle is present given three integers representing the sides of a triangle. The key component of the solution consisted of an *if-then-else-if* statement. Finally, candidates were required to trace through a segment of C code to determine its output and to modify the C code to give a different set of output.

In Part (a), the majority of candidates did not obtain full marks since they did not clearly explain white space.

In Part (b), most candidates were able to define the term *debugging* using their own words. The concept of debugging seems to be well understood among candidates.

In Part (c), most candidates had a general concept of indentation being associated with the appearance or presentation of a program. However, the basic idea of maintenance was not present in their responses.

In Part (d), the responses indicated that candidates lacked the ability to properly sequence code in order to achieve the required results. Thus, for example, the check for equilateral triangle should have preceded the check for isosceles triangle. Many candidates were unable to maintain the use of correct syntax throughout their programs. Program code often lacked logical operators in the condition part of the nested *if* statement and several candidates opted to use a series of *if* statements to avoid the use of a nested *if* statement. However, this resulted in incorrect code since one *if* statement would flow into another one. It should also be mentioned that some candidates gave their responses in C++. Candidates need to use the C language in the present Computer Science syllabus.

In Part (e), a number of candidates lacked the ability to trace through the *for* loop so they were unable to generate the output. Some responses ignored the inner *for* loop indicating that these candidates did not understand nested loops. Of those who correctly traced the required output, only a few were able adjust the loop to generate the required output in Part (e) (ii). Again, candidates need to spend more time writing and tracing through programs in order to answer questions like these.

Paper 03 – School-Based Assessment

In general, performance on the Internal Assessment was very good. In Unit 1, there was a slight decline in the performance on the Internal Assessment compared to 2010. The average mark was sixty-eight per cent compared to seventy per cent in 2010. However, there was a significant decline in the performance on the Internal Assessment in Unit 2 compared to 2010. The average mark was fifty-seven per cent compared to sixty-six percent in 2010. Nevertheless, in both units, the average mark was an improvement on the corresponding 2009 average.

Many students `included documentation which was not required for the Unit 1 SBA including structure charts, Gantt charts, data-flow diagrams, questionnaires and entity-relationship diagrams. Many of the projects contained flowchart algorithms; however, system flowchart symbols were incorrectly used instead of programming flowchart symbols.

There were many instances where full marks were awarded for problem descriptions that were either superficial or overly detailed.

Many students did not provide enough detail in the narratives when describing a solution to the problem identified.

Generally, the presentation of pseudocode algorithms showed improvement in logic compared with previous years and were also easy to follow for many of the projects. However, many students provided a pseudocode algorithm that was identical to their C programs. In some projects, the algorithms were written using incorrect logic. For example, variables were often not initialized before being used and repetition statements did not show variables being incremented correctly.

Students often provided programs containing logical steps that were not included in their algorithms.

Some students made use of alternative programming languages such as C++ and Pascal for writing their programs. Schools and students are reminded that C is the language of the Computer Science syllabus.

Programs were often not printed directly from the C development environment and this made it difficult to properly verify the documentation and modularization of the source code. Many students did not make use of comments and did not indent their C code.

Teachers should ensure that programming projects sufficiently cover items in the syllabus such as the use of arrays and record structures as well as the manipulation of external data files.

Trace tables were used in many instances but these were not required for program testing. Program testing by many students often did not show the use of normal, extreme and erroneous data. The test results were often described but not shown by means of screen images. The screen images are required to verify that the program produces the output as described in the report.

Candidate projects should make use of the section headings given in the Criteria for Marking the Internal Assessment in the syllabus. Many projects often did not include any section headings.

The use of GOTO statements should not be encouraged in structured programming.

Storage media such as CDs should not be included with the submission of the samples.

Care should be taken when preparing the SBA document to ensure the following:

- The table of contents is accurate
- Pages are numbered
- Text size, font style and line spacing are legible for reading and moderation.

Teachers should also ensure that each sample has a completed CSCI1-5 form detailing the marks awarded for the assessment criteria.

UNIT 2 – Further Topics in Computer Science

Paper 01

Performance on the 45 multiple-choice items on this paper produced a mean of 58 out of 90 with candidates' scores ranging from 18 to 86.

Paper 02 – Essay Questions

Section A – Data Structures

Question 1

This question tested candidates' knowledge of abstract data types (ADTs) and their implementation. It required candidates to use the Stack ADT to remove the element at the bottom of a stack. The question also examined candidates' knowledge of manipulating ADTs to achieve a desired outcome. In addition, the question tested candidates' understanding of the implementation of a circular queue as well as their understanding of the bubble sort algorithm.

Part (a) required candidates to define the term *abstract data type* (ADT) and to explain how the Stack ADT is implemented. Most candidates had some knowledge of the ADT but were unable to define it properly. In Part (a)(ii), many candidates listed the operations of the Stack ADT but did not explain how the stack would be implemented in code (e.g. by using an array and a *top* pointer and manipulating these data items in the *push* and *pop* operations).

In Part (b), many candidates did not recognize that this was a problem to be solved using the operations of the Stack ADT (e.g., *push*, *pop*, *isEmpty*). Consequently, they wrote unnecessary code to manipulate the internal data of the stack. Thus, they ended up writing code to push and pop the elements from the stack which was not required.

Performance on Part (c) was generally poor. It was clear that many candidates did not understand how a circular queue operates. Some candidates were able to perform the insertions correctly. However, when an element was deleted, many candidates simply shifted the remaining elements one position to the left. In some responses, the size of the array increased or decreased when elements were inserted or deleted, respectively. It should be mentioned that a circular queue operates on an array of fixed size by maintaining pointers to the first and last elements in the queue. Elements are not shifted when inserting or deleting another element.

The quality of the responses for Part (d) was also generally poor. It seems that the majority of candidates did not have a clear understanding of the bubble sort algorithm. Several candidates described the first pass correctly but did not go on to explain what happens after the first pass. They simply concluded their responses by saying that the algorithm is repeated until the entire array is in sorted order. But, getting the

array in sorted order is the purpose of the sort algorithm in the first place. Some candidates wrote C code in their responses. However, the question did not require the writing of code.

Question 2

This question tested candidates' knowledge and understanding of linked lists as well as their ability to implement the selection sort algorithm in C. The question also required candidates to demonstrate their understanding of what happens in the linear and binary search algorithms.

In Part (a), most of the linked list diagrams drawn were incorrect. Many candidates gave tables instead of the standard representation of a linked list (a set of nodes pointing to each other with a special pointer to the top of the list and a terminator symbol at the last node). Of those who drew a diagram containing nodes with each one pointing to the next node in the linked list, many did not did not use a terminating symbol at the last node in the list. In terms of the explanation, many candidates described advantages of the storage characteristics of a linked list instead of simply describing its main features.

In Part (b), the implementation of the selection sort algorithm was poorly done. Candidates demonstrated weaknesses in storing values in the array at the beginning of the algorithm, in writing the nested *for* loops properly (especially with regard to the initial and terminal values of the loop variables), in swapping elements in the array; they generally did not know how the selection sort algorithm works.

The responses for Part (c) were generally good. However, many candidates did not clearly state the conditions under which the linear search algorithm would terminate. Several candidates also did not properly describe what happens in each execution of the loop in the binary search algorithm.

This question required a significant number of narrative responses from candidates. The level of English was very poor and it was often difficult to decipher what a candidate was trying to say. Many candidates did not answer the questions directly, but tended to write down everything they knew on the specific topic. The level of programming was also very poor indicating that candidates have not developed their programming skills sufficiently in Unit 1 to master the syllabus items in this module.

Section B – Software Engineering

Question 3

This question tested candidate's understanding of various software engineering concepts as well as their ability to draw a Level-0 dataflow diagram corresponding to a given narrative.

In Part (a), candidates were required to explain the waterfall approach to system development. Candidates were expected to identify the stages of the waterfall approach and explain how one stage fed into the subsequent stage. However, very few candidates associated the waterfall model with the systems

development life cycle (SDLC) even though this is one of the most popular approaches for the SDLC. The minority of candidates who made the link did not identify the phases of the waterfall approach.

In Part (b), candidates were required to describe four tasks which must be performed during the design phase of systems development (e.g. architectural design, interface design, data structure design and algorithm design). Only a few of the candidates who attempted this question were able to describe these tasks.

Part (c) required candidates to draw a Level-0 dataflow diagram for a given scenario involving a mail order company. Most candidates attempted this question. However, it seems that many of them were not equipped with "drawing tools". Consequently, many of the diagrams were drawn free hand and ended up being very ugly and unsightly for advanced proficiency candidates. Also, on many occasions, the standard symbols for drawing data flow diagrams were not used.

Question 4

This question tested candidates' knowledge of the techniques used for fact-finding during the analysis phase of systems development as well as the deliverables of the analysis phase. It also tested candidates' understanding of the use of CASE tools in software development as well as their ability to use entity-relationship models for data modelling.

Part (a) (i) was well done overall. Most candidates were able to identify three fact-finding techniques such as interviews, observation and questionnaires. In contrast, Part (a) (ii) was poorly done by most candidates. It was clear that many candidates did not understand the term *deliverables* and as a result only a small percentage were able to accurately state two deliverables.

In Part (b), the responses were generally poor. It was clear that most candidates did not know what a CASE tool is and consequently, their responses for both parts of the question were poor.

Part (c) was fairly well done overall. However, many candidates did not use the correct symbol to denote an attribute of an entity and consequently lost marks for drawing the attributes. Also, candidates had difficulty accurately identifying the many-to-many relationships in the scenario (e.g. a lecturer is hired by many departments and a department hires many lecturers). Many candidates drew lines connecting entities (to represent relationships) but did not label the lines or give the cardinality of the relationships. Several candidates also used non-standard notations to represent cardinality. In the future, candidates are advised to use a standard for their entity-relationship models (such as the Unified Modelling Language or the more dated Chen's notation).

Section C – Operating Systems and Computer Networks

Question 5

This question tested candidates' knowledge and understanding of various networking concepts such as network media, network configuration, networking devices, transmission media and access to data and resources on a network

Some parts of the question were answered fairly well by many candidates. However, the responses to the other parts were often very poor.

Part (a) was not answered well by most candidates. Most of them were unable to correctly identify and explain a difference between an analogue signal and a digital signal. Some rather interesting responses stated that an analogue signal is understood by computers while a digital signal is not.

Most candidates answered Part (b) correctly. However, some candidates gave incorrect answers such as telephone, radio and fax machines. This suggests that these candidates may not be familiar with the concept of transmission media.

In Part (c), some candidates gave good responses by explaining the layout of a client-server network with the aid of a labelled diagram. However, some candidates did not understand what the client-server model is and instead gave answers related to the peer-to-peer model or to network topology such as star or ring.

Part (d) (i) tested candidates' knowledge of network connectivity devices such as bridges, repeaters, routers, and gateways. Some candidates described network *types* instead of network *devices*, while a few described devices that are used in a network such as UPS, server. A few candidates also described a network topology such as star, ring, or bus. This suggested that candidates did not know what network connectivity devices were or did not know the difference between network *topology* and network *devices*.

Part (d) (ii) tested candidates' ability to apply knowledge of transmission media to a specific situation. Responses from candidates revealed that some of them did not know what transmission media is and were thus unable to answer the question correctly. Some candidates even gave responses like cell-phones, server and pager.

Part (d) (iii) tested candidates' knowledge of the different ways that users may be granted access to resources on a network. Many candidates confused the granting of access to resources on the network with preventing access to a network and incorrectly gave responses such as encrypting files and using firewalls. Some candidates also stated that access can be granted by using the Internet, an intranet or an extranet.

Question 6

This question tested candidates' understanding of various networking and operating system concepts. Performance was generally fair.

In Part (a), candidates were often unable to properly or clearly distinguish between a batch processing system and a multi-user system. For example, some candidates stated that in a batch processing system, only one user used the system at a time. The examples provided in Part (a) (ii) and Part (a) (iii) were often not representative enough of the type of system.

The responses for Part (b) were generally good with most candidates giving answers such as passwords and encryption.

The responses for Part (c) were very poor. It is clear that most candidates do not understand the concept of an interrupt and the interrupt processing mechanism.

The responses for Part (d) were also poor. Many candidates did not recognize that the diagram represented a deadlock situation and gave interesting answers such as 'hold ups', 'event interrupt', 'round robin' or even 'parallel processing'.

In Part (e), many candidates were unable to completely describe how the round-robin scheduling algorithm works. It also seemed that weaker candidates had no idea of what was involved in the algorithm.

PAPER 03 – School-Based Assessment

For the Unit 2 SBA, students are expected to choose a problem for which a software solution exists and then develop the software using software engineering techniques. In particular, they are expected to demonstrate the tools and techniques used in the analysis of the software to be developed. They are then expected to design, code and test their software using appropriate techniques.

Performance on the SBA was generally fair. It was observed that the projects submitted by Students were deficient in various aspects and were still being awarded high marks. Teachers also need to become more closely involved in the supervision of the projects.

Generally, most students chose appropriate topics for the SBA. The topics chosen were relevant to the level of the students and the specific objectives of the syllabus. Diagrams were generally done well; however, they were very inconsistent and did not flow (as they should) from one to the other. The treatment of the topics by students was adequate. A small percentage was comprehensive though some tended to be superficial. The reports were also generally well presented.

Students should put more effort in following the layout of the SBA given in the syllabus. This enables students to clearly identify which parts of the SBA are being addressed by which sections of the report.

Teachers should instruct students to use the requirements in the mark scheme as sub-headings within their reports. This ensures that students will have a section corresponding to each requirement.

Teachers need to advise students to restrict the scope of their SBA Reducing the scope of the project or focusing on a single aspect will give students a better chance of completing what is set out in their problem definition.

SBA reports should be well secured and bounded to avoid the mixing up or loss of parts of the reports submitted.

Teachers should not assume that marks are given for things not printed. Only visual/printed elements can be marked so anything not included in printed form within the SBA will not receive marks. Teachers also need to ensure that each candidate's code is *printed* and included in the SBA report. A soft copy of a candidate's code is not accepted.

Teachers should ensure that students clearly state the problem being tackled in the problem definition. Simply stating that a system is manual or paper-based is insufficient. A manual or paper-based system is not in itself a problem, as stated by many students. The fact that it may be time consuming, error prone, tedious and so on are problems that may exist in a particular manual or paper-based system. Not all manual or paper-based systems suffer from these problems; thus, students should identify the problems clearly in order to be awarded full marks.

In describing the techniques used for analysis of the problem, many students did not say why the techniques were relevant.

It should be noted that a few teachers and students continue to confuse the terms *functional requirements* and *non-functional requirements*. Special attention should be paid to the listing of functional requirements as this was generally done poorly.

Students need to extend their system structuring diagram beyond the elements of the main menu in order to be awarded full marks for this requirement.

Students seldom included justifications for their choice of user interface. Also, some of them were confused as to the difference between menu-driven and command-driven interfaces.

Students in general left out the report design; teachers should ensure that this is included and not assume that marks will be awarded based on screen shots.

Test plans were not done well in general. Students should include an exhaustive set of data to be tested which includes all forms of data entry. Many students only tested the menu selections. Expected results and actual results obtained from testing should be included in tests. Tests should also include normal, abnormal and extreme data.

In most cases, the functionality of the program written was poorly described and there were no screen shots of the working program producing the functionality described in the reports. The scope of the programs was often too large; as a result, the stated functionality seldom matched the actual programs submitted. Teachers should encourage students to develop programs that focus on simpler, more specific problems from the start. This would allow students to realistically develop what is proposed early in the documentation.

There were a few cases where students used the old Unit 2 syllabus to prepare their SBA. As a result, irrelevant information was submitted. Teachers should ensure they are using the syllabus marked **Effective for examinations from May/June 2009.**

RECOMMENDATIONS

Students 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. Students 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 students on the Computer Architecture and Organization module can be handled by adequate coverage of the syllabus items. This module does not involve computer programming. Students 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 SBA, teachers need to provide better feedback to students. The SBA 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. It should also be mentioned that the SBA provides an invaluable opportunity to gain hands-on experience with the items in the Computer Science syllabus which ultimately leads to better performance on the written papers. Thus, the SBA should be viewed not as a goal in itself but as a framework for mastering the contents of the syllabus.