INVOCAB CURRICULUM SUPPORT MANUAL

INFORMATION TECHNOLOGY

Developed by Kevin Feveck

Table of Contents

INVOCAB

Information Technology Curriculum Support Manual

Rationale

Computer Science and Information Technology is one of the fastest growing fields in many countries. The Department of Labor projects that in the U.S, the number of jobs in Computer and Information Technology will grow by about 12% from 2014 – 2024 due to increased cloud computing technologies, large scale collection of data (Big Data) and the increasing connectivity of devices to the Internet (Internet of Things), with careers in these fields having some of the highest median wages.[1] It is no surprise that there is a large demand for skilled workers in the fields of Computer Science and Information Technology in the U.S, as it is where some of the largest tech companies in the world are based.

It is a fair assumption to make that the Caribbean region will also soon see the need for skilled workers in these fields as the use of technology continues to proliferate in areas such as education, medicine, and even agriculture. It is therefore imperative that education systems in the Caribbean position themselves such that students are exposed at an early age to the subject area so that the students can build a solid foundation and develop skills necessary to perform well at University level should they choose to pursue these careers.

Problem Areas

The problem areas listed in this section target specific areas in the CSEC syllabus in order to improve student performance in the related areas in the examination. These areas were chosen after reviewing the CSEC I.T Subject Reports over the last 6 years (May/June 2010 – Jan 2016) for both the January and May/June examination periods.

[1] https://www.bls.gov/ooh/computer-and-information-technology/home.htm

The May/June 2010 examination was the first exam based on the updated syllabus, which combined the previous I.T General and I.T Technical syllabi.

Across the years, there are several recurring problems noted by examiners, particularly in the following areas:

- Number representation and manipulation
- Data verification and validation
- Programming implementation
- Use of flowcharts, and meaning of symbols
- Familiarity with PASCAL programming language, such as syntax
- Programming concepts such as control structures, data types
- Definitions of terms, listing advantages/disadvantages, listing use cases
- SBA uniqueness
- Submitting only soft copies of SBAs, improper marking schemes, and differing marking schemes when there are multiple teachers
- Lack of assessment of all skills as outlined by CSEC guidelines for SBAs

It should be noted that student performance is much worse in questions related to number representation/manipulation and programming when compared to other topics, with the majority candidates scoring poorly. Across all the years reviewed, candidate performance on the questions on programming is always reported as poor. These areas should be given more attention by teachers and instructors, and as a result three (3) of the five (5) listed potential problem areas are directed at addressing performance in these areas (Math Skills, Critical Thinking and Problem Solving, and Programming Knowledge).

The Subject Reports were used as the main contributor for determining and outlining problem areas. The following is a list of potential problem areas that may need to be addressed. The following appear in no particular order:

1. Math Skills
2. Critical Thinking and Problem Solving skills
3. Programming Knowledge/General Knowledge
4. Emerging Topics/Issues in the Computer Science/ Information Technology field
5. SBA component of course

The following is a brief outline and rationale behind each area.

1) Math Skills

Competency in math [1] and confidence in one's ability [2] are linked to good performance Computer Science/Information Technology, especially in programming related tasks. In addition to findings that link good math skills to excelling in programming tasks, there is a math component in the CSEC I.T syllabus which accounts for the focus on developing student's Math Skills.

In some cases, examiners noted candidates having difficulty solving programming questions that relied on computing mathematical functions such as averages, with candidates coming up with the wrong formula. Candidates also struggled with number representations and manipulations such as one's and two's complement, number base conversion, and addition of these number representations.

2) Critical Thinking and Problem Solving skills

Good critical thinking and problem solving skills are a cornerstone of a good programmer. Difficulties observed by examiners with the programming type questions across a majority of candidates in the examinations, point to a weakness in the development of these skills in candidates. Candidates have been noted to struggle with core concepts such as control structures, arrays and declarations, which points to a lack of knowledge in these key programming concepts.

A lack of knowledge of these key concepts would also indicate a lack of knowledge of how to use them, hence the need to develop candidates' critical thinking and problem solving skills so that they know when these concepts are applicable and how to use them.

3) Programming Knowledge

As a direct consequence of the last problem area, programming knowledge should also be concentrated on. Candidates were noted to have struggled with program implementation, data types, knowledge of syntax, identifying errors in code excerpts, and defining terminology, for example, 'runtime error' or 'object code.' Examiners reported that candidates did not seem familiar with the PASCAL programming language and there were reports that a large number of candidates were unable to write PASCAL. These problems may be a result of lack of practice writing code and a general discomfort with programming.

4) General Knowledge/ Emerging Issues and Topics in the Field

Candidates were observed to have trouble with terms and definitions, but this was not limited to programming areas. These difficulties extended to other areas including communications and security. Candidates often had trouble with explanation type questions discussing advantages/disadvantages or similarities, giving the meaning of acronyms, or talking about the uses of various technologies. This points to a lack of knowledge of the subject matter, as these questions typically do not have a mathematical or programming component attached to them; they merely test the candidate's general knowledge of the field.

Additionally, in the few years where candidates were asked questions about jobs in the field, CSEC reported that many candidates had difficulty. Candidates could not link job tasks to job titles. It may be the case that candidates are not aware of the many career paths in Computer Science and Information Technology. While not a core area of the syllabus with respect to skills and competencies, it is very important that candidates are educated about the career opportunities that are available to them in the field, particularly those careers that will be important to the Caribbean region.

Another issue, which is not directly related to the syllabus, is Women in Computer Science (CS). There are efforts worldwide to bridge the gender gap particularly in STEM fields, which are majority males. In 2016 in the United States, women accounted for 25% of the workforce in computer and mathematical sciences [3]. Looking at UWI Student Statistics from the years 2013 to 2016, the average number of students enrolled in Computer Science (full-time and part-time) was 165 and the average number of female students was 37, only 22%. Looking at the statistics for students enrolled in Information Technology (full-time and part-time), out of an average of 218 total enrolled students 70 are female, or 32%. A further look at those statistics over those 3 years reveals that 29.5% of all students enrolled in the university are enrolled in either in the Faculty of Engineering or Faculty of Science and Technology, and of those students about 11% are in Computer Science or Information Technology programs. [4] [5] [6] With jobs in technology being in heavy demand worldwide due to advances in cloud computing and Internet technologies, it is important to educate students about the importance of these careers to the future development of the Caribbean region with the hope that it inspires interest for them to pursue these careers.

5) SBA component

The SBA component has several of the same problems being reported every year. These problems include lack of uniqueness in the SBA, improper marking schemes as compared to the guidelines set forth by CXC, poorly organized SBA submissions with regards to presentation and sectioning of the documents submitted, and failure to send a single common marking scheme when there are multiple instructors. These problems point to poor effort from both the students as well as the instructor.

Objectives

The instructor is the most important teaching tool, and the performance of the student is heavily dependent on the ability of the instructor to uphold the standards set out by CXC guidelines with respect to assessment of skills and competencies, and the quality of SBA submissions. This problem area is linked to all others. By assisting the instructor with teaching strategies, classroom activities, and key areas to focus on, this manual seeks to improve student performance through improving instruction.

The objectives of this manual are as follows:

- To identify main problematic areas relating to syllabus objectives as observed from CSEC subject reports
- To identify secondary problematic areas from supporting literature on general problems faced in the area of Computer Science and Information Technology
- To provide teaching strategies and classroom activities to instructors for various topics in the CSEC I.T syllabus that have been identified as problem areas
- To provide instructors with information and tips to educate students about, and encourage them to pursue, careers in Computer Science and Information Technology

Problem Area 1: Math Skills

Topics	CSEC Syllabus Reference
Manipulation of Units of Storage	Section 1, objective 4
Number Representation	Section 1, objective 9
Manipulation of Number Representations	Section 1, objective 9

OBJECTIVES

- To define the different units of storage and demonstrate how to convert between them
- To define the various number representations and explain their uses
- To demonstrate how to convert between various number representations
- To demonstrate how to perform arithmetic operations in a given number representation

INSTRUCTIONAL FOCUS

The instructor should focus primarily on having students perform practical exercises for this problem area. Regular practice will help students to feel more confident, so the instructor should incorporate short quizzes weekly, or even at the beginning of every class. As this teaching area is heavily dependent on use of math skills, instructors should look for signs of math anxiety in students and attempt to mitigate these attitudes in the classroom. [7] Math anxiety is a real concern when teaching these areas, and instructors must be careful to not go through the material too quickly, or give students problems that are too difficult to work with. Students' success rate influences the number of problems they attempt, and attempting more problems generally improves student scores [8]. The advancement of the student's confidence in his/her abilities and encouraging positive attitudes toward math is just as important as developing math skills and competencies.

SUGGESTED ACTIVITIES

Please refer to "AS and A Level Computer Science: Teacher pack" in the Resources section for more detailed activities, instructional videos, and guides to teaching content.

- Exercise worksheets (individual and group work)

- Short quizzes
- Basic and simple mental exercises on conversion and arithmetic
- Matching exercises on terminology/definitions
- Have students create their own questions (with answers), and exchange their questions with peers

Common Issues

- Lack of frequent practice of arithmetic skills in the classroom
- Confusion between various number representation systems, or units of storage
- Students may feel intimidated by math, have low self-efficacy in math skills

Background Information

Units of Storage

bit – smallest unit of information, represented as either a 1 or 0

byte – a group of 8 bits

nibble – half of a byte, or 4 bits

prefix – kilo, mega, giga, tera. There should be a distinction made between a binary prefix, which uses powers of 2, and a decimal prefix, which uses powers of 10. For example, a 500-gigabyte hard drive typically means there are 500,000,000,000 bytes (decimal prefix) a 2-gigabyte stick of ram means there are 2,147,483,648 bytes (binary prefix). Usually, a lowercase letter is used for decimal notation (kB) where as an uppercase letter is used for binary notation (KB)

Prefixed bytes	Number (binary prefix)
Kilobyte (KB)	1024 bytes
Megabyte (MB)	1024 kilobytes
Gigabyte (GB)	1024 megabytes
Terabyte (TB)	1024 gigabytes

Word – a fixed-size unit of data made up of n-bits (e.g. 32 bits on a 32-bit architecture).

Word size – the size of the word as defined in the system, determined by the architecture of the processor, and is typically 32-bit or 64-bit.

Data/Number Representations

Binary – base 2 number representation, uses values 0-1

Octal – base 8 number representation, uses values 0-7

Hexadecimal – base 16 number representation, uses values 0-9 and characters A-F to represent 10-15

One's complement – one's complement of a binary number is defined as the value obtained by inverting every bit of the original binary number (e.g. the one's complement of 0110 is 1001)

Two's complement – two's complement of a binary number is defined as the value obtained by inverting every bit of the original binary number and adding 1 (e.g. the two's complement of 0110 is 1010)

Binary Coded Decimal (BCD) – a system each digit of a decimal of a number is represented in n-bits, typically 4-bit or 8-bit. There are also special representations used to represent the sign of the number (e.g. the BCD representation of 49 using a 4-bit system is 0000 0100 1001, the first nibble represents the sign of the number, positive, the second represents '4' and the third represents '9')

Sign and Magnitude – a system where 1 bit, usually the most significant bit, is used to represent the sign of the number (0 for positive, 1 for negative) and the remaining bits are used to represent the value. (e.g. 10000110 represents -6 in an 8-bit representation. Here the most significant bit is 1 representing a negative number, and the remaining 7 bits represent 6)

Representing characters/ASCII – there are several character representation schemes, with the two most popular being ASCII (American Standard Code for Information Interchange) and Unicode. ASCII uses 8-bit patterns but there are only 128 ASCII characters encoded, so the most significant bit is not used. Unicode uses 16-bit patterns.

Suggested Teaching Strategies and Activities

When introducing these concepts for the first time, the instructor should make it clear to the student the purpose of what is being taught. (For example, one's and two's complement are number systems used to represent negative signed numbers in binary representations, and is useful for performing binary arithmetic.)

To overcome math anxiety in students and build the student's confidence in his/her mathematical abilities should be practiced and evaluated frequently. Instructors should consider having students regularly complete short exercises in number representation and manipulation, such as converting between representations or performing arithmetic. The instructor should go through these exercises as a class so that students receive feedback immediately.

Instructors should be wary of how attitudes to math, and effective strategies to address these attitudes, differ between boys and girls. For example, it has been suggested that girls benefit more than boys from problems tailor-made to their ability, and making errors in private [8]. In general, instructors should try as much as possible to adapt the difficulty of problem sets to the individual, and individual ability. This may present a challenge for larger classes. However, this can be addressed by tailoring problem sets to cater for students with observed poor performance. It is important to ensure a high success rate across the class before increasing difficulty of problems.

It is suggested that instructors implement primarily active learning and cooperative learning teaching strategies. Allowing students to actively take part in the learning process and engage their peers will foster development of critical thinking skills, an important area in I.T and Computer Science. Students may lack an interest while learning material in this area, and this may in turn impact their willingness to do adequate work to develop these skills. Game-based strategies are therefore encouraged so that student's are motivated to work on problems. See "CS Unplugged – Binary Numbers" in Further Reading section for a game-based exercise in teaching binary numbers. To alleviate difficulties with students learning at different rates, instructors can use software/applications where students progress individually, at their own pace. (see Binary Challenge in Further Reading section).

Resources

Resource Name: AS and A Level Computer Science: Teacher pack

Topic: Data types

Resource link: http://www.ocr.org.uk/Images/253757-data-types-topic-exploration-pack-teacher-pack.pdf

Further Reading

Topic	Resource	Link

Binary numbers	"The Socratic Method: Teaching by Asking Instead of by Telling" by Rick Garlikov	http://www.garlikov.com/Soc_Meth.html
	MathManiaCS lesson on binary numbers	http://www.mathmaniacs.org/lessons/01-binary/index.html
	CS Unplugged – Binary Numbers	http://csunplugged.org/binary-numbers/
	Binary Challenge (App – iOS/Android)	http://www.binarychallengeapp.com/
Two’s Complement	Cornell University Lecture notes	https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html
	Arithmetic operations on binary numbers	https://www.doc.ic.ac.uk/~eedwards/compsys/arithmetic/
	Two’s complement subtraction example (video)	https://www.youtube.com/watch?v=vfY7bN_3VKw
Data Representation (number and text)	Computer Science Field Guide (Teacher version)	http://csfieldguide.org.nz/en/teacher/chapters/data-representation.html

Expected Outcomes

The student should feel confident in his/her ability to do the following:

- Define all terminology listed in this section
- Given a quantity in one unit of storage, convert to another unit of storage
- Given a number represented in a specified number representation, convert to another number representation
- Be able to evaluate an arithmetic expression in a given a number representation

Problem Area 2: Critical Thinking and Problem Solving

Topics	CSEC Syllabus Reference
Understanding the concept of Problem Solving	Section 2, objectives 1,5,6
Decomposing a problem	Section 2, objectives 2,3
Solving the problem	Section 2, objectives 4,7,9
Testing the solution to a problem	Section 2, objective 8

OBJECTIVES

- To demonstrate how to formulate a problem statement
- To describe the problem solving process
- To demonstrate how to break down a problem and identify its significant parts
- To show how to apply the problem solving process
- To define variables, constants, the various data types, and to demonstrate when to use each
- To define and describe key symbols in flowcharts and key words in pseudocode, and identify their uses
- To demonstrate how to construct pseudocode and flowcharts to represent an algorithm
- To demonstrate how to use trace tables to verify an algorithm's correctness

INSTRUCTIONAL FOCUS

The instructor should plan lessons around presenting scenarios or cases presented as word problems to the students, which is solved as a class or in groups. The examples should be as close to real life problems as possible, adjusted for the CSEC level. The instructor should try as much as possible to allow students to arrive at answers themselves, guiding them along the way, by asking questions, for example.

SUGGESTED ACTIVITIES

- Word-problem exercises
- Group activities

Common Issues

- Students have difficulty reasoning about a problem. This may be because of a lack of internalization of the skills/knowledge required to solve this problem, or it may be something more basic such as the student not understand what the question is asking (English comprehension skills)
- When given code by an instructor, students may "copy and paste" code for use as solutions in future problems, without understanding the code.
- Students may be confused about the difference between pseudocode and actual code, as noted in CSEC IT reports. (It is important that instructors explain that the term "pseudo" means fake, or resembling, so that students distinguish it from an programming language)
- When constructing flowcharts, students may not draw shapes properly, or misuse shapes/symbols

Background Information

Problem statement – a clear and concise description of the issues that need to be solved

Significant parts of a problem – input, process, storage, output

Input – What data, if any, is required at the start of the problem?

Process – What steps must be taken, with input or otherwise, to solve the problem?

Storage – What are the data types of the input? What are the constants and the variables in this problem? What must be stored during the Process stage? What are the data types of the output?

Output – What information should come out of this process?

Variables – a stored value that may change during the execution of the program, or needs to be entered at run time

Constant – a stored value that does not change during the execution of a program

Data types – Integer, floating point, character, literal

> *Integer* – whole number values. e.g. 100, 0, -34
>
> *Floating-point* – real number values. e.g. 1.0, 2.56, -0.53
>
> *Characters* – strings or letter values. e.g. “apple”, ‘b’, “@TT”
>
> *Literal* – source code representation of a fixed value, for example assigning the variable x = 6, the number 6 is referred to as an integer literal.

Characteristics of Algorithms – finiteness, definiteness, input, output

> *Finiteness* – algorithms must have a finite number of steps, after which it terminates
>
> *Definiteness* – algorithms must have an unambiguous flow of control from one step to another, and each step is clearly and precisely defined
>
> *Input* – algorithms have 0 or more inputs, but never an infinite number of inputs
>
> *Output* – algorithms have at least 1 or more outputs, but never an infinite number of outputs

Suggested Teaching Strategies and Activities

Problem-based learning strategies should be implemented in this module as it focuses on developing critical thinking skills. Students will be more engaged during lessons and are likely to ask more questions in this teaching environment. The instructor should note that this teaching strategy is generally more time consuming in the classroom as opposed to more traditional methods, and more effort is required to tailor lesson plans to this teaching strategy. When choosing word problems, instructors do not necessarily need to be confined to the I.T/programming domain. Word problems from other domains, such as Mathematics or Physics may be borrowed as long as the instructor clarifies any domain specific knowledge and provides formula where needed, and the problems are at a suitable level of difficulty. Mathematical word problems, such as computing averages, sums or areas of shapes, should especially be used where possible so that students can develop math skills in tandem.

A small but relevant distinction between project-based learning and problem-based learning should be made: project-based learning generally involves working on more complex and larger problems than problem-based learning. It is important to present more simple problems than fewer complex problems to the students, so that problem solving and critical thinking skills have increased opportunities to develop. In the further reading section, see “Tips on problem based learning” for a brief guide to problem based learning, and “Polya’s Problem Solving Techniques” for a guide on what students should be doing to solve problems and develop critical thinking skills. See also “Problem Solving Basics” by R. Pasko in resources section for worked examples complete with pseudocode and flowchart solutions. See “Group Activities” document included in the activities folder of accompanying material.

Resources

Resource Name: Graded Math Word Problems (varying difficulties, with Solutions)

Topic: Problem Solving

Resource link:

Grade 10: http://www.analyzemath.com/high_school_math/grade_10/problems.html

Grade 9: http://www.analyzemath.com/middle_school_math/grade_9/problems.html

Grade 8: http://www.analyzemath.com/middle_school_math/grade_8/problems.html

Grade 7: http://www.analyzemath.com/middle_school_math/grade_7/problems.html

Grade 6: http://www.analyzemath.com/middle_school_math/grade_6/problems.html

Resource Name: 81 Fresh & Fun Critical-Thinking Exercises

Topic: Problem Solving

Resource link:
http://www.mathematicshed.com/uploads/1/2/5/7/12572836/81_fun_critical_thinking_activities.pdf

Resource Name: Simple Programming Problems (varying difficulty)

Topic: Problem Solving

Resource link: https://adriann.github.io/programming_problems.html

Additional Notes: in this module, students should only concern themselves with coming up with pseudocode/flowcharts to solve the problems, not write actual code. This will be addressed in the next problem area.

Resource Name: Problem Solving Basics by R. Pasko

Topic: Problem Solving

Resource link: http://www.cs.iit.edu/~cs100/ProblemSolving.pdf

Resource Name: How to Solve It: A New Aspect of Mathematical Method

Topic: Problem Solving

Resource link: http://math.hawaii.edu/home/pdf/putnam/PolyaHowToSolveIt.pdf

Further Reading

Topic	Resource	Link
Tips on problem based learning	5 Keys to Rigorous Problem Based Learning	https://www.edutopia.org/video/5-keys-rigorous-project-based-learning
Tips on Problem Solving	Polya's Problem Solving Techniques	https://math.berkeley.edu/~gmelvin/polya.pdf
Flowcharts	Materials on flowcharts	http://www.bbc.co.uk/education/guides/z3bq7ty/revision/3
		https://www.slideshare.net/devaashish1/algorithms-and-flowcharts

Expected Outcomes

The student should feel confident in his/her ability to:

- Answer explanatory type questions pertaining to the problem solving process and program design
- Formulate a problem statement given a text description of a problem
- Decompose a problem statement and extract significant parts from a given problem statement

- Given a description of an algorithm, construct pseudocode or a flowchart to represent the algorithm
- Given pseudocode/a flowchart, test and validate the algorithm by using trace tables, showing input, output, and any associated working or processing
- Combine skills throughout the problem solving process cycle to solve a problem presented as a scenario (from formulating the problem statement straight through to testing and validating the algorithm)

Problem Area 3: Programming Knowledge

Topics	CSEC Syllabus Reference
Programming Knowledge and Terminology	Section 3, objectives 1,2,3,4,11
Storing, Retrieving and Manipulating Data	Section 3, objectives 5,6,7,9
Control Structures	Section 3, objectives 8

OBJECTIVES

- To describe high and low level languages and their differences
- To describe the five generations of programming languages and their differences
- To define the steps associated with program implementation
- To define common programming terminology
- To demonstrate how to declare and manipulate variables and constants
- To demonstrate how to use control structures

INSTRUCTIONAL FOCUS

The instructor should regularly quiz students on the definitions they are expected to know. These quizzes should be short, and administered regularly, similarly to what is described for Problem Area 1: Math Skills. Instructors should teach some of the history of computing so that students know and understand how technology developed from the first computers to modern day. Instructors should show how programming languages evolved based on needs.

For the programming aspect, students should begin writing and reading code as early as possible so they can get to know the syntax of the language. However, instructors should ensure students are comfortable with, and understand, the programming concepts/algorithms presented as pseudocode before demonstrating an implementation in actual code. It is advised that instructors pair these lessons with concepts taught in Problem Area 2: Critical Thinking and Problem Solving so that students see how algorithms (pseudocode and flowcharts) are used to develop actual code, and see how the problem solving process works holistically. Students should begin writing code on paper or text editors first so that they can learn the syntax of the programming language, and instructors should always correct even the most minor of mistakes, such as leaving out semicolons.

SUGGESTED ACTIVITIES

- Short quizzes
- Given an algorithm, filling in code excerpts
- Fill out trace tables
- Given lines of code and an algorithm, put lines of code in correct order (puzzle-based programming)
- Finding syntax, or logical errors in code excerpts
- Matching exercises (programming generation to description)
- SCRATCH Programming

Common Issues

- Students may be confuse pseudocode with actual code and vice versa
- Students may have problems using, and combining, logical operations (AND, OR, NOT) and mathematical operators (<, >, etc.) when using IF statements
- Issues understanding how to use basic looping structures. Students may have trouble with using For loops to iterate through arrays, for example when doing linear search. (Off-by-one error)
- Students may have trouble using While loops, coming up with correct conditions to ensure that loop will terminate eventually

Background Information

Low-level language – low-level languages are those which map very closely to a processor's instruction set, providing very little or no abstraction. Low-level languages generally refer to Machine Code or Assembly language. Low-level languages generally run very quickly when compared to high-level languages and require less memory to store and execute, but they are not as portable as the instructions are on the underlying hardware architecture. Low-level language requires direct management of memory in a program. First and Second generation languages are low-level languages

High-level language – high-level languages provide layers of abstraction, which make it closer to human language. These are languages such as PASCAL, C, Java, Python, Prolog, etc. High-level languages are much more portable due to the layers of abstract it provides, which removes concern for the details of the underlying hardware. High-level languages often offer support for memory management, some automatically like Java (garbage collection), but some languages such as C require the programmer to expressly allocate and de-allocate memory. High-level languages are generally less efficient than low-level languages as it requires more memory to store and execute a program, and there are intermediary steps that convert the high-level language to a low-level language that the processor can execute. Third, Fourth, and Fifth Generation languages are high-level languages

Generations of programming languages

First generation – Machine code, originally entered through panels at the front of the machine, first generation languages are machine code. They do not have to be compiled or assembled. It is typically written in binary, however it may also be in hexadecimal, octal, etc. and be translated to binary. First-generation languages are very efficiently because it is executed directly, however it is very difficult to debug if there is an error because it is difficult to interpret by human programmers. First generation languages are considered low-level.

Second generation – Assembly language, called such because the codes need to be converted to a machine-readable form by a process called "assembly". Assembly language is much easier to read by a programmer than machine code, as the codes resemble words such as "move", "store", "add", etc. However, similar to first-generation languages, it is not portable as it the language is specific to the underlying hardware architecture of the processor. Second generation languages are considered low-level.

Third generation – This generation of programming languages (PASCAL, C, Java, C++, etc.) are much more portable and more programmer friendly because they provide multiple layers of abstraction between the programmer and the underlying hardware. These are general-purpose languages and most provide structured programming concepts such as subroutines, block structures (‘begin’ and ‘end’, parentheses, etc.) and looping structures. They are much easier to read, write and debug because it is closer to human language, but they must be compiled or interpreted to machine code before they can be executed, so they are less efficient and require more memory when compared to low-level languages.

Fourth generation – This generation of programming languages (R, SAS, SQL, Adobe ColdFusion) operates at a higher level of abstraction, with the programmer specifying what must be done without necessarily explicitly specifying how to do it. These languages typically operate on a much more vast collection of data, or have a specific use or domain. These uses include report generation, database management, web development, and complex mathematical operations. They are even closer to human language than 3rd generation languages.

Fifth generation – This generation of programming languages (Prolog, Mercury) is based on logic programming or constraint programming has very specific applications such as in Artificial Intelligence research or proof solving. Programmers provide a set of constraints instead of providing an actual algorithm to find solutions.

General Terminology

Source code – the human readable language implementation of a program, including comments. It is usually just plain text, and must be compiled or so that it can be executed. Alternatively, it may be interpreted and executed on the fly.

Compile – to translate human readable high-level languages to low-level languages such as machine code. Compilation takes the source code and generates object code.

Linking – to take the object file(s) generated by a compiler and combine it into an executable file

Executing – to run a program via its executable file

Testing – to verify the correctness of the output of a program by using known or pre-computed input-output pairs

Test data – data that is used in the testing phase. The output of this data is known before hand and is used to verify correctness of the implementation

Debugging – the act of locating and correcting errors in the source code of a program

Syntax Error – an error that arises when the compiler does not understand a statement in the source code. Source code containing syntax errors will not compile.

Logic Error – an error in the source code that may or may not cause the program to crash, but produces erroneous behaviour or output. The only way to detect logic errors is by testing. Source code containing logical errors will compile.

Run-time Error – an error that occurs during execution that causes the program to crash. Runtime errors may be result of a logical error in the source code, for example division by zero, or the result of some other failure such as running out of memory.

Dry run – when programmers verify code manually before running it, by going through the code line by line to look for errors.

Elementary Data Types

Integers – this data type stores whole number values (e.g. -2, 0, 10). Integer types in a programming language typically can only represent values within a certain range due to the fixed number of bytes used to store them, for example -2,147,483,648 to 2,147,483,647 for a 4-byte (32-bit) integer in C.

Reals – also called floating-point values, this data type stores real number values (e.g. -2.5, 0.02, 100.3434). Real, or float, types in a programming language typically can only represent values within a certain range due to the fixed number of bytes used to store them, for example 1.2 x 10^{-38} to 3.4 x 10^{38} for a 4-byte (32-bit) float in C.

Characters – these are strings or letter values. e.g. “apple”, ‘b’, “@TT”

Boolean – special True or False values, however, they are not explicitly present in all languages, for example C uses 1 or 0 instead. (not in syllabus, however mentioned for completeness)

Suggested Teaching Strategies and Activities

To get students to understand and remember the definitions in this section, such as the various generations of programming languages, the main teaching strategy should be lectures, or YouTube or other videos in place of lecture, accompanied by regular quizzing on the material. The quizzes should be short so that they can be done quickly in the classroom, and the instructor can grade them and give students feedback with a minimal turn-around time. The main purpose is to get students to learn and remember through repetition. These quizzes can be in the form of fill in the blank, multiple-choice questions, matching terms with definition type exercises, etc. Instructors are encouraged to make a pool of such questions and recycle them to make this process easier on them.

Instructors should show how programming languages evolved based on needs. Going from Machine code-> assembly language ->general purpose languages ->4th generation domain specific languages, each progression has more layers of abstractions so programmers can read and write code much easier. Instructors can implement alternative activities such as having students role-play as programmers from the various eras, and have the students describe what their experience coding would have been like. Another activity is having students create posters of a timeline of programming languages. Students can accompany descriptions of the era with pictures of computers from each era, popular jobs from that era, or highlight prominent members of the field from that era.

For the programming aspect of this module, instructors should use problem-based learning, puzzle-based learning, and pair programming teaching strategies as these are shown to be effective. [9] Instructors should try to simplify terms as much as possible for students. For example, when explaining the concept of variables, instructors can show that it is merely a container for holding an object. Furthermore, these 'containers' have a type, and these types describe what objects are allowed to be stored in these containers. While more focus should be placed on students understanding the logical aspect of programming, according to CXC reports, students have problems with proper programming syntax. To address this, students should write and read actual code as much as possible so that they learn the syntax of the language through exposure. Therefore it is advised that students write code using pen and paper, or on a basic text editor instead of using an Integrated Development Environment (IDE). Using an IDE before the student knows the syntax of the language may be harmful, as advanced IDEs will often pick up the slack through features such as autocomplete (for example, automatically adding semicolons to end of line), etc.

Resources

Resource Name: Simple Programming Problems (varying difficulty)

Topic: Programming

Resource link: https://adriann.github.io/programming_problems.html

Resource Name: Pascal Programming for Beginners

Topic: Programming

Resource link: http://www.pascal-programming.info/lesson1.php

Resource Name: Pascal Tutorial (with online Pascal compiler built-in to lessons)

Topic: Programming

Resource link: https://www.tutorialspoint.com/pascal/index.htm

Resource Name: A practical introduction to Pascal programming language

Topic: Programming

Resource link: http://giara.unavarra.es/pdfs/APITPPL.pdf

Further Reading

Topic	Resource	Link
History of computing	A History of Computer Programming languages	https://cs.brown.edu/~adf/programming_languages.html
	A Brief History of Computers	http://www.jeremymeyers.com/comp/
	A history of Programming Languages (CS 313 Middlebury)	http://www.cs.middlebury.edu/~schar/courses/cs313-s17/docs/HistoryProgLang.pdf
Teaching Strategies for programming	An Overview of Computer Programming Teaching Methods	http://archive.ceciis.foi.hr/app/index.php/ceciis/2011/paper/view/431/238
	SCRATCH Programming by MIT Media lab	https://scratch.mit.edu

Expected Outcomes

The student should be confident in his/her ability to:

- Give a description of low level languages, high level languages, and the various generations of programming languages
- Compare and contrast high and low level languages, and compare and contrast among the generations of programming languages
- List steps associated with program implementation and briefly describe each step
- Explain programming terminology listed in this section
- Write code to declare variables (including arrays), read and write to variables, and use them to do simple operations such as arithmetic or linear search (arrays)
- Use and implement control structures
- Given an algorithm in pseudocode or flowchart form, implement the algorithm in a programming language

Problem Area 4: General Knowledge/Emerging Issues and Topics in the Field

Topics	CSEC Syllabus Reference
General Terminology and Knowledge	Section 1, objectives 1-15 Section 4, objectives 1-7 Section 5, objectives 1-7 Section 6, objectives 2,13,14 Section 7, objectives 1,2 Section 8, 1-4
Issues in the Field	N/A
Jobs in the Field	Section 4, objectives 8,9

OBJECTIVES

- To define all terms and acronyms contained in CSEC syllabus for objectives listed above under topic "General Terminology and Knowledge", and give descriptions of functions or uses of such items where applicable
- To describe career paths for Information Technology and Computer Science, and to describe the roles and duties of jobs in these field listed in the CSEC syllabus
- To outline the jobs/career paths that are currently trending/in high demand in the U.S., U.K. (and other leading markets in Information Technology and Computer Science) which are relevant to Trinidad and Tobago, and the wider Caribbean
- To identify issues in computing, such as, but not limited to:
  1. Being aware the laws governing online behavior and use of the Internet
  2. Using social networking sites, and the Internet safely

3. Lack of women and diversity in Computer Science/Information Technology fields

INSTRUCTIONAL FOCUS

The instructor should administer short quizzes regularly on the definitions and terms listed in "General Terminology and Knowledge" so that students can remember them through repeatedly being tested on them, similarly to what is advised in Problem Areas 1 and 3.

When educating student's about jobs in the field, the instructor should also inform students how they can pursue careers in these fields for example, by showing them entry requirements for Bachelors and Masters programs at local tertiary education institutions. The more important focus here is inspiring interest in pursuing careers in the field, rather than simply memorizing job titles and their associated roles and duties.

Instructors should refer to materials listed in the Further Reading section to present to students such the laws that govern the Internet in Trinidad and Tobago, or how to use the Internet safely especially when it comes to social networking sites. While this area does not relate to specific objectives in the CSEC syllabus, it is important that students learn how to use the Internet safely and responsibly in modern times, and what consequences may await those who do not.

SUGGESTED ACTIVITIES

- Scenarios of dangers that exist online
- Short quiz
- Matching exercises for job titles and job roles
- Scenarios of real world uses of I.T/Computer Science
- Social media/Online safety videos from YouTube

Common Issues

- Students may not see how the "issues" section is relevant because it does not relate to CXC syllabus, (students must have motivation/reason for learning something to take it serious)
- Students may lack interest in learning how to use the web safely, and understanding the laws that govern the web
- Students may want to pursue in careers in Information Technology or Computer Science are dissuaded by poor performance, or low self-confidence

Background Information

For terms and definitions in topic "General Terminology and Knowledge" please refer to listed sections and objectives of CSEC syllabus, as they are too numerous to list.

Given the prevalence of smartphones, laptops and other devices that can access among students, and the availability of the Internet it is important to educate them on how to be safe and responsible on the Internet. It is important that students follow basic safety tips such as making profiles private/increasing privacy settings, not using the same password for all online accounts, turn off location services on smartphones and other devices and not share location data with social networking sites.

As reported above, looking at the most popular in the region and largest local tertiary education institution, U.W.I, enrollment in Computer Science and Information Technology is relatively low. The numbers of women who enroll in these programs are even lower, 22% and 32% on average over the last 3 years. It is important to encourage a diverse workforce in all sectors, and as a result it is important to encourage students, especially young girls, to look at careers in these fields.

Suggested Teaching Strategies and Activities

Instructors should use lecture style strategies for teaching definitions and terms in "General Terminology and Knowledge." Please see "Glossary" in the CSEC syllabus. Instructors should use active learning strategies when presenting materials on jobs and careers in I.T/Computer Science. It is important that the students discuss and engage with the instructor to allow any natural inquisitions they have to aid teaching. Instructors should encourage students to find job ads in the newspapers or online, having them pay specific attention to the roles/duties and the require skills in the job descriptions, and then discussing this with their peers and the instructor. These jobs can be local, regional or from other international countries, however, instructors should guide students to look at job ads that have relevant locally or regionally, or will be in the near future.

Instructors can also implement active learning strategies when discussing issues that are in the field, for example online piracy and the penalties according to the law in T&T. Other issues such as online bullying, identity theft, grooming should be discussed. Instructors can present possible scenarios to students and discuss with them what the "red flags" in the situation are, and the associated dangers and consequences.

Resources

-

Further Reading

Topic	Resource	Link
Issues in the Field – Laws governing online behaviour	TT Computer Society-compilation of ICT-related Laws and Policies in Trinidad and Tobago	http://www.cs.tt/articles/ict-related-laws-and-policies-in-trinidad-and-tobago/
Issues in the field-Laws governing online behavior	Proposed Cyber Crime bill 2017	http://www.ttparliament.org/legislations/b2017h15.pdf
Jobs in the field	UWI MSc Data Science (Dept. of Computer Science and Information Technology)	http://ai.lab.tt/
	UWI MSc Computer Science	https://sta.uwi.edu/fst/dcit/academics_pg_cs.asp
Being safe online	Staying Safe on Social Networking Sites	https://www.us-cert.gov/ncas/tips/ST06-003 http://www.ncpc.org/topics/internet-safety/social-networking-safety https://kids.usa.gov/parents/online-safety/six-tips-for-keeping-teens-safe-on-social-media/index.shtml https://www.dhs.gov/sites/default/files/publications/Social%20Media%20Guide_3.pdf https://www.youtube.com/watch?v=hqezbib5qpQ https://www.youtube.com/watch?v=6jMhMVEjEQg https://www.youtube.com/watch?v=c4sHoDW8QU4
Issues in the field-lack of women in the field	Rear Admiral Dr. Grace Murray Hopper	https://www.computerhope.com/people/grace_hopper.htm http://www.cs.yale.edu/homes/tap/Files/hopper-story.html
	Dr. Margaret Bernard	https://sta.uwi.edu/fst/dcit/margaret.bernard.asp https://sta.uwi.edu/rdifund/MargaretBernard.asp

	Head of Department, Computer Science and Information Technology, UWI	
	Dr. Kim Mallalieu	http://www.eng.uwi.tt/depts/elec/staff/kmallalieu/kmallalieu.php http://www.niherst.gov.tt/icons-1/women-in-science/kim-mallalieu.html

Expected Outcomes

The student should be confident in his/her ability to:

- Give a brief explanation of terms in the CSEC syllabus outlined in topic “General Terminology and Knowledge”
- Given a job title from objectives listed in CSEC syllabus as outlined in topic “Jobs in the Field”, give a brief description of roles and duties of said job
- Use the internet more safely and responsibly
- Pursue a career in Information Technology, Computer Science, and related fields should they choose to

Problem Area 5: School-Based Assessment

OBJECTIVES

- To ensure all students submit a proper CSEC SBA, according to CSEC guidelines
- To submit a single marking per school scheme to CSEC
- To ensure both hard and soft copies of SBAs are submitted, properly bound and organized

INSTRUCTIONAL FOCUS

N/A

SUGGESTED ACTIVITIES

N/A

Common Issues

- Lack of uniqueness among SBAs/Too many similar SBAs from students of same school
- Improper mark schemes for SBA
- Where there are multiple teachers, multiple mark schemes are submitted instead of a single, common mark scheme
- Generally, teachers are not rigorous enough with marking/mark schemes
- In word processing sections especially, not all skills are being assessed, CSEC guidelines for marking should be followed
- In programming sections, students removed semicolons from PASCAL code and submitted it as pseudocode

Background Information

N/A

Suggested Teaching Strategies and Activities

The instructor should plan SBAs at the start of the school year, ideally at the start of Form 3 or 4. As the instructor works through the CSEC syllabus, the associated parts of the SBA should be worked on in tandem. This will ease pressure on both the student and the instructor.

Resources

Resource Name: Guidelines for the Conduct of the School-Based Assessment

Topic: School-Based Assessment

Resource: CSEC Syllabus, page 23

Further Reading

N/A

Expected Outcomes

- Students should have a SBA that was graded according to a suitable marking scheme based on CSEC guidelines
- In cases where there are multiple teachers/classes submitting SBAs from a single school, a single, common Marking Scheme will be used and submitted to CSEC
- SBAs should be submitted soft and hard copy, with hard copy properly organized, ordered and bound

Works Cited

1 Fan, T. S., & Li, Y. C. *Is math ability beneficial to performance in college computer science programs*. Journal of National Taipei Teachers College, 2002.

2 Sáinz, M., & Eccles, J. *Self-concept of computer and math ability: Gender implications across time and within ICT studies*. Journal of Vocational Behavior, 2012.

3 Foundation, National Science. *Science and Engineering Indicators*. National Science Foundation, 2016.

4 University of the West Indies, St. Augustine. *Student Statistics*. UWI , 2013/2014.

5 University of the West Indies, St. Augustine. *Student Statistics*. UWI, 2014/2015.

6 University of the West Indies, St. Augustine. *Student Statistics*. UWI, 2015/2016.

7 Maloney, Erin A., and Sian L. Beilock. Math anxiety: Who has it, why it develops, and how to guard against it. *Trends in cognitive sciences*, 16, 8 (2012), 404-406.

8 Jansen, Brenda RJ, et al. The influence of experiencing success in math on math anxiety, perceived math competence, and math performance. *Learning and Individual Differences*, 24 (2013), 190-197.

9 Mohorovičić, Sanja, and Vedran Strčić. An Overview of Computer Programming Teaching Methods. In *XXII Central European Conference on Information and Intelligent Systems* (2011), 1-6.