GIRLS' HIGH SCHOOL
INFORMATION TECHNOLOGY

Form 5 Academic Year: September 2025 – July 2026
Term 1 – 13 weeks

♦ *SECTION 7: PROBLEM-SOLVING AND PROGRAM DESIGN*

GENERAL OBJECTIVES

On completion of this Section, students should:

1. develop the cognitive skills to solve problems; and,
2. *develop competence in* applying a structured approach to solving problems on the computer.

SPECIFIC OBJECTIVES	CONTENT
Students should be able to:	
1. outline the steps in problem-solving;	*Steps in problem-solving:* (a) define the problem; (b) propose and evaluate solutions; (c) determine the most efficient solution; (d) develop the algorithm; and, (e) test and validate the solution.
2. *use the divide-and-conquer approach to decompose large everyday problems into smaller tasks;*	*Basic treatment of the structured approach for solving complex problems.* ***Note:*** *It is not necessary to give a detailed treatment of the approach. Simple illustrations can be provided to help students recognize that most problems involve multiple tasks and that they should understand how to approach such problems in a structured manner.*
3. define a problem by decomposing it into its significant components;	The components are: input; process; and output. A defining diagram *(IPO Chart)* may be used to delineate the components.
4. distinguish between variables and constants;	Variables as an area of storage whose value can change during processing; the value of a constant never changes. Data types: integers, floating point (real), characters, *Boolean, string*.

5.	explain the concept of algorithms;	Definition of algorithms. Characteristics: finite number of steps, precise, unambiguous, flow of control from one process to another, terminate.
6.	represent algorithms in the form of *flowchart and pseudocode;* and,	Use of flowchart symbols: input/output, process, decision, directional arrows, start/stop. Pseudocode – Use of read, input, store, write, *print,* output, display, *conditional branching (if-then, if-then-else, nested conditions); loops (for, while, repeat).* Use of relational operators: <, >, =, <=, > =,< >. Logical operators: AND, OR, NOT*; use of truth tables.* Arithmetic operators: +, –, *, /, *MOD, DIV.*
7.	test algorithms for correctness.	*Desk checks/dry run:* construction and use of trace tables to verify results. Trace tables consist of variable names (identifiers) as column headings and values in the cells, one row for each pass.

13 WEEKLY SCHEDULE OUTLINING OBJECTIVES AND ACTIVITIES FOR THE CSEC INFORMATION TECHNOLOGY 2020 SYLLABUS, SECTION 7: PROBLEM-SOLVING AND PROGRAM DESIGN WITH A SUGGESTED ALLOCATION OF 3 HOURS PER WEEK.

WEEK	WEEKLY OBJECTIVE	ACTIVITIES
1	Understand the Program Design Life Cycle and Problem Definition.	Introduce the Problem-Solving Phase (Define, Analyze, Propose, Develop Algorithm, Test) and the Program Implementation Phase. Discuss the characteristics of a well-defined problem (precise, unambiguous). Use non-computing examples (like a recipe or directions) to introduce steps and sequence.
2	Decompose a problem using the IPO Chart.	Explain and practice breaking down problems into Input, Process, and Output using the IPO (Input-Process-Output) chart. Students define and chart simple problems (e.g., calculating the area of a rectangle).
3	Grasp the concepts of Variables and Constants.	Define variables (names, changing values) and constants (fixed values). Discuss the importance of meaningful variable names. Practice identifying variables and constants in simple mathematical problems.

4	Identify and use fundamental Data Types.	Introduce and explain common data types: Integer (whole number), Real/Float (decimal), Character, and String (text). Students practice assigning appropriate data types to variables from problem statements.
5	Define and develop the basic Algorithm.	Define an algorithm and its key attributes (precise, unambiguous, finite, logical sequence). Introduce the Sequence control structure. Write simple algorithms in plain English/pseudocode for sequential tasks (e.g., calculating an average of three numbers).
6	Represent simple algorithms using Pseudocode.	Introduce standard pseudocode structure (Start, Stop, Read, Display, Calculate/Assign). Convert simple sequential English algorithms from Week 5 into formal pseudocode.
7	Represent simple algorithms using Flowcharts.	Introduce standard flowchart symbols (Terminal, Input/Output, Process, Flow Lines). Convert the simple sequential pseudocode from Week 6 into flowcharts. Practice drawing neat and correct flowcharts.
8	Apply the Selection (Decision) Control Structure.	Introduce and explain the IF...THEN...ELSE and CASE (or nested IF) structures. Students write pseudocode and flowcharts for problems involving a single decision (e.g., check if a number is positive).
9	Apply complex Selection Control Structures.	Practice algorithms involving multiple decisions using nested IF statements or the CASE structure (e.g., determining a student's grade based on a numerical score).
10	Apply the Repetition (Loop) Control Structure: Counted Loop.	Introduce the concept of a loop and the counted loop (e.g., FOR loop). Develop algorithms using a counted loop for a fixed number of repetitions (e.g., print a message ten times).
11	Apply the Repetition Control Structure: Conditional Loops.	Introduce the conditional loops (WHILE loop and REPEAT...UNTIL loop). Differentiate between pre-test and post-test loops. Develop algorithms to process an unknown number of items using a sentinel value (e.g., calculate the sum of numbers until 0 is entered).
12	Test and Validate Algorithms using Trace Tables.	Explain the purpose of testing and validation. Introduce and practice using a trace table to manually step through an algorithm (sequential, selection, and repetition) to ensure the logic produces the correct output for test data.
13	Review and Evaluate Alternative Solutions.	Review all problem-solving steps and control structures. Present a single problem and have students propose two or more alternative algorithms. Discuss and evaluate which solution is the most efficient (fewer steps, clearer logic).