Scope and sequence: IB Diploma CS

Sha Tin College, class of 2027

Time period	Lesson focus
September, October	B2: Programming
Mid term break	
November, December	B1: Computational thinking B3: Object oriented programming
Winter break	
January, February, March	B4: Abstract data types A3: Databases
April break	
May, June	Year 12 timed assessments Internal assessment lessons Submit Internal Assessment draft (Criteria A, B, C)
Summer break	Submit Internal Assessment draft (Criteria D, E)
September, October	A1: Computer fundamentals A4: Machine learning
Mid term break	
November, December	A4: Machine learning (continued) A2: Networks Submit Internal Assessment final
Winter break	
January, February, March	Mock examinations Case study Exam revision
April break	
May, June	External examinations

Detailed scope and sequence

PROGRESS DRAFT

Week of	Topic	Lesson	Title	Mapping	Teaching & learning
18/08	B2: Programming	1	Hello world		1. Install VS Code 2. Install Github Desktop 3. Create a Github account 4. Create a Github repo 5. Add me as a collaborator (Github user: `paulbaumgarten`) 6. Practice uploading to your Github repo 7. Checking the automated testing ● Solve Me First (Function, arithmetic)
18/08	B2: Programming	2	Numeric types & operations	B2.1.1 Construct and trace programs using a range of global and local variables of various data types. ● Data types: Boolean value, char, decimal, integer, string	Exercise 1: Temperature Tracker https://pbaumgarten.com/docs/ib-compsci-2027/b2.html#exercise-1-temperature-tracker Exercise 2: Spell checker https://pbaumgarten.com/docs/ib-compsci-2027/b2.html#exercise-2-spell-checker Exercise 3: Maze navigator https://pbaumgarten.com/docs/ib-compsci-2027/b2.html#exercise-3-maze-navigator Exercise 4: Frequency counter https://pbaumgarten.com/docs/ib-compsci-2027/b2.html#exercise-3-maze-navigator Exercise 5: Robot instructor https://pbaumgarten.com/docs/ib-compsci-2027/b2.html#exercise-5-robot-instructions Leetcoode problems ● #1 - Two Sum (Nested loops, lists) ● #412 - Fizz Buzz (Conditionals, loops) ● #9 - Palindrome Number (Integer/str conversion) ● #125 - Valid Palindrome (String cleaning) ● #88 - Merge Sorted Arrays (Lists, loops)
18/08	B2: Programming	3	String types & operations	B2.1.2 Construct programs that can extract and manipulate substrings. ● Writing of programs that accurately identify and extract substrings from given strings, demonstrating the ability to perform various manipulations, such as altering, concatenating or replacing	
25/08	B2: Programming	4	Arrays & lists	B2.2.2 Construct programs that apply arrays and Lists. ● One-dimensional (1D) arrays, two-dimensional (2D) arrays, ArrayLists in Java ● One-dimensional (1D) Lists and two-dimensional (2D) Lists in Python	

				• Add, remove and traverse elements in a dynamic list	• #26 - Remove Duplicates from Sorted Array (List processing) • #38 - Count and Say (String processing) Hackerrank problems • Simple Array Sum (Loops, summing values) • Compare the Triplets (Lists, conditionals) • A Very Big Sum (Large number handling) • Diagonal Difference (2D lists) • Plus Minus (Counting, floats) • Staircase (patterns, printing) • Mini-Max Sum (Sorting, arithmetic) • Birthday Cake Candles (Max value counting)
25/08	B2: Programming	5	Arrays & lists		
25/08	B2: Programming	6	Sequence & selection	B2.3.1 Construct programs that implement the correct sequence of code instructions to meet program objectives. • The impact of instruction order on program functionality • Ways to avoid errors, such as infinite loops, deadlock, incorrect output B2.3.2 Construct programs utilizing appropriate selection structures. • Must include: if, else, else if (Java), elif (Python), to execute different code blocks based on specified conditions • Selection structures with or without Boolean operators (AND, OR, NOT) and/or relational operators (<, <=, >, >=, ==, !=) to control program flow effectively	
25/08	B2: Programming	7	Iteration	B2.3.3 Construct programs that utilize looping structures to perform repeated actions. * Types of loops, including counted loops and conditional loops, and appropriate use of each type * Conditional statements within loops, using Boolean and/or relational operators to govern the loop's execution	
01/09	B2: Programming	8	Iteration		
01/09	B2: Programming	9	Functions & modularisation	B2.3.4 Construct functions and modularization. • Functions to define reusable blocks of code with different inputs	

				• Modularization to create well-structured, reusable and maintainable code • The principles of scope (local versus global) • The benefits of code modularization, applying this concept to various programming scenarios	
01/09	B2: Programming	10	Exception handling & debugging techniques	B2.1.3 Describe how programs use common exception handling techniques. Potential points of failure in a program must include unexpected inputs, resource unavailability, logic errors. The role of exception handling in developing programs Exception handling constructs that effectively manage errors must include try/catch in Java, and try/except in Python, along with the finally block. B2.1.4 Construct and use common debugging techniques. Debugging techniques may include trace tables, breakpoint debugging, print statements and step-by- step code execution.	Exercise 1: Student grades calculator Exercise 2: Simple ATM Simulator
08/09	B2: Programming	11	Stacks	B2.2.1 Compare static and dynamic data structures. • The fundamental differences between static and dynamic data structures, including their underlying mechanisms for memory allocation and resizing • The advantages and disadvantages of each type in various scenarios, considering factors such as speed, memory usage, flexibility B2.2.3 Explain the concept of a stack as a “last in, first out” (LIFO) data structure. Must include fundamental operations such as push, pop, peek and isEmpty How stack operations impact both performance and	• Leetcode 20 Valid Parentheses • Leetcode 155 Min stack • Hackerrank Balanced brackets • Advent of code 2021 day 10 Syntax Scoring

				memory usage An appropriate stack for a specific problem	
08/09	B2: Programming	12	Queues	B2.2.4 Explain the concept of a queue as a “first in, first out” (FIFO) data structure. Must include fundamental operations such as enqueue, dequeue, front and isEmpty How queue operations impact both performance and memory usage An appropriate queue for a specific problem	• Leetcode 1700 Number of students unable to eat lunch • Hackrank Queue using Two Stacks • Advent of code 2019 Day 5 part 1 Sunny with a Chance of Asteroids
08/09	B2: Programming	13	Big O	B2.4.1 Describe the efficiency of specific algorithms by calculating their Big O notation to analyse their scalability. • The time and space complexities of algorithms and calculating Big O notation • Algorithm choice based on scalability and efficiency requirements	
08/09	B2: Programming	14	Search algorithms	B2.4.2 Construct and trace algorithms to implement a linear search and a binary search for data retrieval. • The differences in efficiency between different methods of linear and binary search • Use of search technique based on efficiency requirements—for example, searching a database for a sorted/indexed list of names to find a phone number, versus searching by the number to identify the name	• Leetcode 704 Binary search • Hackerrank Ice Cream Parlor • Leetcode 35 Search insert position
15/09	B2: Programming	15	Search algorithms		
15/09	B2: Programming	16	Sort algorithms	B2.4.3 Construct and trace algorithms to implement bubble sort and selection sort, evaluating their time and space complexities.	• Hackerrank 30 days of code Day 20: Sorting • Leetcode 75 Sort Colors • Leetcode 88 Merge sorted array

				• The time and space complexities of each algorithm, denoted by their respective Big O notations • The advantages and disadvantages of each algorithm in terms of efficiency across various data sets	• Advent of code 2020 day 5 Binary Boarding
15/09	B2: Programming	17	Sort algorithms		
22/09	B2: Programming	18	Recursion (HL)	B2.4.4 Explain the fundamental concept of recursion and its applications in programming. (HL only) • The fundamentals of recursion and its advantages and limitations • The utility of recursion in solving problems that can be broken down into smaller, similar sub-problems • Recursive algorithms, including but not limited to quicksort • The limitations of recursion, including complexity and memory usage • Situations that best suit the use of recursion, including fractal image creation, traversing binary trees, sorting algorithms B2.4.5 Construct and trace recursive algorithms in a programming language. (HL only) • Simple, non-branching recursive algorithms in programming only	• Hackerrank 30 days of code Day 9 Recursion 3 • Leetcode 509 Fibonacci number • Leetcode 49 Permutations • Advent of code 2019, day 6 Universal Orbit Map • Leetcode 733 Flood fill (DFS) • Leetcode 695 Max area of Island (DFS) Quicksort exercise Sudoku exercise
22/09	B2: Programming	19	Recursion (HL)		
22/09	B2: Programming	20	Recursion (HL)		
22/09	B2: Programming	21	File processing	B2.5.1 Construct code to perform file-processing operations. • Programs that manipulate text files	Log file parser, CSV file parser, search sort the file, CSV analytics

				• Opening a sequential file in various modes (read, write, append) • How to read from and write to files, append data to an existing file, and close a file once operations are completed • Classes for Java users may include Scanner, FileWriter, BufferedReader. • Functions for Python users may include open(), read(), readline(), write(), close().	
29/09	B2: Programming	22	Programming scenarios		
29/09	B2: Programming	23	Programming scenarios		
29/09	B2: Programming	24	Programming scenarios		
OCTOBER MID-TERM BREAK					
13/10	B2: Programming	25	Programming scenarios		
13/10	B2: Programming	26	Programming scenarios		
13/10	B2: Programming	27	Programming scenarios		
13/10	B2: Programming	28	Review		
20/10	B2: Programming	29	Review		
20/10	B2: Programming	30	Assessment		
20/10	B3: OOP	1	Introducing OOP	B3.1.1 Evaluate the fundamentals of OOP. • Model real-world entities using OOP concepts: classes, objects, inheritance, encapsulation, polymorphism	

				• The advantages and disadvantages of using OOP in various programming scenarios	
27/10	B3: OOP	2	Designing classes	B3.1.2 Construct a design of classes, their methods and behaviour. • Classes and their methods, based on application requirements • The use of unified modelling language (UML) class diagrams to represent class relationships, attributes and methods, to aid effective software design and planning	
27/10	B3: OOP	3	Instantiating objects	B3.1.4 Construct code to define classes and instantiate objects. • How to define classes and create objects from those classes • The role of constructors in initializing an object's state, setting initial values for its attributes to define its condition or characteristics at the time of creation	
27/10	B3: OOP	4	Encapsulation	B3.1.5 Explain and apply the concepts of encapsulation and information hiding in OOP. • The principles of encapsulation and information hiding • Apply access modifiers such as private and public • Controlling access to class members • The importance of limiting access to maintain the integrity and security of an object's state	
27/10	B3: OOP	5	Statics & non-statics	B3.1.3 Distinguish between static and non-static variables and methods.	

				• The differences between static and non-static variables and methods, including their usage and scope • When to use instance variables instead of class variables, and how to apply these concepts effectively in code	
03/11	B3: OOP	6	Programming scenarios		
03/11	B3: OOP	7	Programming scenarios		
03/11	B3: OOP	8	Programming scenarios		
EXPLORER WEEK					
17/11	B3: OOP	9	Inheritance (HL)	B3.2.1 Explain and apply the concept of inheritance in OOP to promote code reusability. • How inheritance enables a hierarchical relationship between parent and child classes • Extending existing classes, utilizing inheritance to reuse and extend functionalities • The impact of inheritance on access to parent class members with different access modifiers (private, public, protected, default)	
17/11	B3: OOP	10	Polymorphism (overriding) (HL)	B3.2.2 Construct code to model polymorphism and its various forms, such as method overriding. • The principle of polymorphism and how it contributes to code flexibility and reusability • How to implement dynamic polymorphic behaviour through mechanisms like method overriding	

				• How to apply static polymorphic behaviour to maximize code efficiency	
17/11	B3: OOP	11	Abstract classes (HL)	B3.2.3 Explain the concept of abstraction in OOP. • The significance of abstraction in the development of modular code fragments • The use of abstract classes to establish common interfaces for sub-classes	
17/11	B3: OOP	12	Composition & aggregation (HL)	B3.2.4 Explain the role of composition and aggregation in class relationships. • How to design objects by leveraging smaller component objects through composition and aggregation • That aggregation implies that the subcomponents can function independently of the aggregating class, while in composition, the subcomponents are tightly coupled and cannot exist outside the aggregating class	
24/11	B3: OOP	13	Design patterns (HL)	B3.2.5 Explain commonly used design patterns in OOP. • The key design patterns such as singleton, factory and observer • The application of design patterns in solving recurring programming challenges	
24/11	B3: OOP	14	Programming scenarios (HL)		
24/11	B3: OOP	15	Programming scenarios (HL)		
01/12	B3: OOP	16	Programming scenarios (HL)		
01/12	B3: OOP	17	Programming		

			scenarios (HL)		
01/12	B3: OOP	18	Exam style questions		
01/12	B3: OOP	19	Exam style questions		
08/12	B3: OOP	20	Assessment		
08/12	B1:Comp thinking	1	Problem specification	B1.1.1 Construct a problem specification. ● The specification of a problem may include a problem statement, constraints and limitations, objectives and goals, input specifications, output specifications, evaluation criteria.	
08/12	B1:Comp thinking	2	Computational thinking concepts & application	B1.1.2 Describe the fundamental concepts of computational thinking. ● Abstraction, algorithmic design, decomposition, pattern recognition B1.1.3 Explain how applying computational thinking to fundamental concepts is used to approach and solve problems in computer science. ● Computational thinking does not necessarily involve programming—it is a toolkit of available techniques for problem-solving. ● Real-world examples may include software development, data analysis, machine learning, database design, network security.	
08/12	B1:Comp thinking	3	Trace flowcharts	B1.1.4 Trace flowcharts for a range of programming algorithms. ● Use of standard flowchart symbols to depict processes, decisions and flows of control ● Standard flowchart symbols: Connector, Decision, Flowline, Input/Output, Process/Operation, Start/End	

				• Flowcharts for execution flow, to track changes in variables and to determine output	
WINTER BREAK					
05/01	B4: ADTs		Intro to ADT (HL)	B4.1.1 Explain the properties and purpose of ADTs in programming. • The core principles of ADTs, including their purpose in providing a high-level description of data structures and their associated operations	
05/01	B4: ADTs		Linked lists (HL)	B4.1.2 Evaluate linked lists. • Lists must include singly, doubly, circular • Sketch of linked lists and implementation of basic operations diagrammatically, such as insertion, deletion, traversal, search • The advantages and disadvantages of using linked lists over other data structures like arrays, particularly in terms of memory utilization and performance B4.1.3 Construct and apply linked lists: singly, doubly and circular. • The basic operations on a linked list, such as insertion, deletion, traversal, search	
05/01	B4: ADTs		Linked lists (HL)		
05/01	B4: ADTs		Linked lists (HL)		
12/01	B4: ADTs		Binary search trees (HL)	B4.1.4 Explain the structures and properties of BSTs. • How binary search trees (BSTs) are used for data organization • Insert, delete, traverse and searching nodes in a BST • Sketching a BST as a tree diagram	

12/01	B4: ADTs		Binary search trees (HL)		
12/01	B4: ADTs		Binary search trees (HL)		
19/01	B4: ADTs		Sets (HL)	B4.1.5 Construct and apply sets as an ADT. ● The fundamental characteristics of sets, including their unordered nature and the uniqueness of elements ● Operations: union, intersection and difference ● Code to check if an element is in a set, to add an element to a set, to remove an element, and to check whether one set is a subset/superset of another set	
19/01	B4: ADTs		Sets (HL)		
19/01	B4: ADTs		Sets (HL)		
19/01	B4: ADTs		Hashmaps (HL)	B4.1.6 Explain the core principles of ADTs. ● High-level description of data structures and their associated operations and purpose ● The underlying mechanics of hash tables, including hashing functions, collision resolution strategies and load factors ● The underlying mechanics of sets to store and manage data ● HashMap and HashSet in Java; dict and set in Python	
26/01	B4: ADTs		Hashmaps (HL)		
26/01	B4: ADTs		Hashmaps (HL)		
26/01	B4: ADTs		Programming		

			scenarios (HL)		
02/02	B4: ADTs		Programming scenarios (HL)		
02/02	B4: ADTs		Programming scenarios (HL)		
02/02	B4: ADTs		Exam style questions (HL)		
02/02	B4: ADTs		Exam style questions (HL)		
09/02	B4: ADTs		Exam style questions (HL)		
09/02	B4: ADTs		Assessment (HL)		
CHINESE NEW YEAR BREAK					
23/02	A3: Databases	1	Database fundamentals	A3.1.1 Explain the features, benefits and limitations of a relational database. ● Features: composite keys, foreign keys, primary keys, relationships, tables ● Benefits of databases: community support, concurrency control, data consistency, data integrity, data retrieval, reduced data duplication, reduced redundancy, reliable transaction processing, scalability, security features ● Limitations of databases: “big data” scalability issues, design complexity, hierarchical data handling, rigid schema, object-relational impedance mismatch, unstructured data handling	
23/02	A3: Databases	2	Schemas and data types	A3.2.1 Describe database schemas. ● Conceptual schema, logical schema, physical schema	

				• Abstract definitions of the data structure and organization of the data at different levels A3.2.3 Outline the different data types used in relational databases. • The importance of data type consistency • The potential effects of choosing the wrong data type	
23/02	A3: Databases	3	Entity relationship diagrams	A3.2.2 Construct ERDs. • The significance of entity relationship diagrams (ERDs) in crafting organized, efficient database designs tailored for specific applications • The relationships between different data entities within a database • The roles of cardinality and modality in defining relationships in ERDs A3.2.4 Construct tables for relational databases. • The relationship between tables using primary keys, foreign keys, composite keys and concatenated keys • The importance of well-defined tables in ensuring data integrity	
02/03	A3: Databases	4	Normalisation	A3.2.5 Explain the difference between normal forms. • First normal form (1NF), second normal form (2NF), third normal form (3NF) • The terms atomicity, unique identification, functional dependencies, partial-key dependencies, non- key/transitive dependencies • Normalization issues can encompass data duplication, missing data, and a range of dependency concerns, including data	

				dependencies, composite key dependencies, transitive dependencies, and multi-valued dependencies. A3.2.7 Evaluate the need for denormalizing databases. ● The advantages and disadvantages of normalizing and denormalizing databases ● Situations where denormalization can enhance performance, particularly in read-intensive applications ● The balance between straightforward query structures and the risk of data redundancy in denormalized schemas	
02/03	A3: Databases	5	Designing 3NF databases	A3.2.6 Construct a database normalized to 3NF for a range of real-world scenarios ● Examples may include library management, hospital management, e-commerce platforms, school management, employee management, inventory management, police crime reporting	Event registration database. Provide a list of fields for students to group into tables based on anticipated relationships. ● Eg: Same event multiple years, how should that be treated? ● Change of address or phone number over time? Keep old or replace one entry? ● Payment information? Marksbook database. E-Commerce database.
02/03	A3: Databases	6	Introducing SQL	A3.3.1 Outline the differences between data language types within SQL. ● Data language types must include data definition language (DDL) and data manipulation language (DML) ● SQL statements to define data structures or to manipulate data	Construct the database for one of the previous models
09/03	A3: Databases	7	SQL update & insert	A3.3.3 Explain how SQL can be used to update data in a database. ● Insert new records (INSERT INTO), modify data (UPDATE SET), remove data (DELETE) The performance implications of updating data in indexed columns, and how indexes might need to	Provide some same data for students to use. create INSERTs with and UPDATES. Start doing SELECTS

				be rebuilt or reorganized following significant data modifications	
09/03	A3: Databases	8	SQL joins	A3.3.2 Construct queries between two tables in SQL. ● Queries must include joins, relational operators, filtering, pattern matching, and ordering data ● SQL commands: SELECT, DISTINCT, FROM, WHERE, BETWEEN, ORDER BY, GROUP BY, HAVING, ASC, DESC, JOIN, LIKE with % wildcard, AND, OR, NOT (note: Syntax may vary in different database systems)	
09/03	A3: Databases	9	SQL aggregate functions (HL)	A3.3.4 Construct calculations within a database using SQL's aggregate functions. (HL only) ● Aggregate functions on grouped data to aid reporting and decision-making ● Aggregate commands: AVERAGE, COUNT, MAX, MIN, SUM	
09/03	A3: Databases	10	Views (HL)	A3.3.5 Describe different database views. (HL only) ● Virtual views and materialized (snapshot) views ● Hiding data complexity, data consistency, independence, performance, query simplification, read-only data or updatable data, security	
16/03	A3: Databases	11	Transactions (HL)	A3.3.6 Describe how transactions maintain data integrity in a database. (HL only) ● The role of atomicity, consistency, isolation and durability (ACID) to ensure reliable processing of transactions	

				• Transaction control language (TCL) commands: BEGIN TRANSACTION, COMMIT, ROLLBACK	
16/03	A3: Databases	12	Alternatives & warehouses (HL)	A3.4.1 Outline the different types of databases as approaches to storing data. • Databases models: NoSQL, cloud, spatial, in-memory • Examples of the use of the database model in real-world scenarios may include e-commerce platforms, geographic information systems (GIS), managed services, real-time analytics, social media platforms, SaaS. A3.4.2 Explain the primary objectives of data warehouses in data management and business intelligence. • The roles of append-only data, subject-oriented data, integrated data, time-variant data, non-volatile data and data optimized for query performance, to ensure efficient data storage and analysis	
16/03	A3: Databases	13	Data mining & distributed databases (HL)	A3.4.3 Explain the role of online analytical processing (OLAP) and data mining for business intelligence. • Data mining techniques must include classification, clustering, regression, association rule discovery, sequential pattern discovery, anomaly detection (note: This links to “A4 Machine learning”). • The uses of the techniques in extracting meaningful information from large data sets A3.4.4 Describe the features of distributed databases.	

				• The need to maintain data consistency in a distributed database • The role of ACID to ensure reliable processing of transactions in distributed databases • Features of distributed databases: concurrency control, data consistency, data partitioning, data security, distribution transparency, fault tolerance, global query processing, location transparency, replication, scalability	
23/03	A3: Databases	14	Using SQL with Python (bonus)		
23/03	A3: Databases	15	Exam style questions		
23/03	A3: Databases	16	Assessment		
APRIL BREAK					
YEAR 12 EXAMS					
	Internal assessment	1	Assessment overview	Advice on project selection Review of exemplar projects Research ideas	
	Internal assessment	2	Submit project proposal		
	Internal assessment	3,4	Criterion A Problem specification	Scenario, context, success criteria	
	Internal assessment	5,6	Criterion B Planning	Initial planning UML, structure diagram, gantt	
	Internal assessment	7-12	Criterion C System overview	1 lesson for each of... UX diagrams Flowcharts UML overview Extras such as case diagram, DFD, networking	

				diagram, ML modeling etc Functional testing Structural testing	
	Internal assessment	13-30	Programming	Self directed programming time	
	Internal assessment	31-33	Criterion D Development documentation and video	Development documentation and video	
	Internal assessment	34-35	Criterion E Evaluation		
SUMMER BREAK					
	A1: Computer fundamentals	1	CPU components	A1.1.1 Describe the functions and interactions of the main CPU components. ● Units: arithmetic logic unit (ALU), control unit (CU) ● Registers: instruction register (IR), program counter (PC), memory address register (MAR), memory data register (MDR), accumulator (AC) ● Buses: address, data, control ● Processors: single core processor, multi-core processor, co-processors ● A diagrammatic representation of the relationship between the specified CPU components	

	A1: Computer fundamentals	2	Primary memory, secondary memory	A1.1.4 Explain the purposes of different types of primary memory.
				
				● Random-access memory (RAM), read only memory (ROM), cache (L1, L2, L3), registers
				● The interaction of the CPU with different types of memory to optimize performance
				● The relevance of the terms “cache miss” and “cache hit”
					Compare read/write speeds for different memory media. RAM disk, SSD, USB portable drive, old HDD?
					
					poe starter:
					
					`import time`
					
					`# Write a file`
					`data = "A" * 100000000  # 100MB of data`
					`start_time = time.time()`
					`with open("test_file.txt", "w") as file:`
					`  file.write(data)`

				A1.1.7 Describe internal and external types of secondary memory storage.
				
				● Internal hard drives: solid state drive (SSD), hard disk drive (HDD), embedded multimedia cards (eMMCs)
				● External hard drives: SSD, HDD, optical drives, flash drives, memory cards, network attached storage (NAS)
				● The scenarios in which the various types of drive are used
					`write_time = time.time() - start_time`
					`print(f"Write Time: {write_time:.4f} seconds")`
					
					`# Read the file`
					`start_time = time.time()`
					`with open("test_file.txt", "r") as file:`
					`  file.read()`
					`read_time = time.time() - start_time`
					`print(f"Read Time: {read_time:.4f} seconds")`

	A1: Computer fundamentals	3	Fetch / decode / execute	A1.1.5 Describe the fetch, decode and execute cycle.
				
				● The basic operations a CPU performs to execute a single instruction in machine language
				● The interaction between memory and registers via the three buses: address, data, control
					Implement a CPU simulator in Python.
					Implement code for a few common operations
					Provide a "memory" that will perform a simple task?
					
					Starting point from poe:
					
					`class CPU:`
					`  def __init__(self):`
					`  self.PC = 0  # Program Counter`
					`  self.IR = None  # Instruction Register`
					`  self.MAR = None  # Memory Address Register`
					`  self.MDR = None  # Memory Data Register`
					`  self.AC = 0  # Accumulator`
					`  self.memory = [10, 20, 30, 40, 50]  # Simulated memory`
					
					`  def fetch(self):`
					`  self.MAR = self.PC`
					`  self.MDR = self.memory[self.MAR]`
					`  self.IR = self.MDR`
					`  print(f"Fetched Instruction: {self.IR}")`
					`  self.PC += 1`
					
					`  def decode(self):`
					`  print(f"Decoding Instruction: {self.IR}")`
					`  # Simulate decoding, can expand with instruction sets`
					
					`  def execute(self):`
					`  print(f"Executing Instruction: Add {self.IR} to Accumulator")`
					`  self.AC += self.IR`
					`  print(f"Accumulator: {self.AC}")`
					
					`cpu = CPU()`
					`for _ in range(3):  # Simulate 3 cycles`
					`  cpu.fetch()`

					cpu.decode() cpu.execute()
	A1: Computer fundamentals	4	GPU (HL)	A1.1.2 Describe the role of a GPU. ● The architecture that allows graphics processing units (GPUs) to handle specific tasks and makes them suitable for complex computations ● Real-world scenarios may include video games, artificial intelligence (AI), large simulations and other applications that require graphics rendering and machine learning. A1.1.3 Explain the differences between the CPU and the GPU. (HL only) ● Differences in their design philosophies, usage scenarios ● Differences in their core architecture, processing power, memory access, power efficiency ● CPUs and GPUs working together: task division, data sharing, coordinating execution	Access a GPU? Train a simple linear regression model, compare the training time with CPU-only vs GPU-capable?
	A1: Computer fundamentals	5	Pipelining (HL)	A1.1.6 Describe the process of pipelining in multi-core architectures. (HL only) ● The instructions fetch, decode, execute ● Write-back stages to improve the overall system performance in multi-core architectures ● Overview of how cores in multi-core processors work independently and in parallel	
	A1: Computer	6	Compression	A1.1.8 Describe the concept of compression.	Implement run-length encoding? Provide a monochrome image (RAW), see how small it can be compressed using RLE.

	fundamentals			• The differences between lossy compression methods and lossless compression methods
				• Run-length encoding and transform coding
					Decompress back to original to ensure no data loss.
					
					Use this as a starter but add use of PIL to load an image.
					
					`# Example 5x5 Image (Black and White)`
					`image = [`
					`  [1, 1, 1, 0, 0],`
					`  [0, 0, 1, 1, 1],`
					`  [1, 0, 0, 0, 1],`
					`  [1, 1, 1, 1, 1],`
					`  [0, 0, 0, 0, 0]`
					`]`
					
					`def rle_compress_image(image):`
					`  compressed = []`
					`  for row in image:`
					`  row_str = ''.join(map(str, row))`
					`  compressed.append(rle_compress(row_str))`
					`  return compressed`
					
					`def rle_decompress_image(compressed):`
					`  decompressed = []`
					`  for row in compressed:`
					`  row_str = rle_decompress(row)`
					`  decompressed.append(list(map(int, row_str)))`
					`  return decompressed`
					
					`# Compress and Decompress the Image`
					`compressed_image = rle_compress_image(image)`
					`print("Compressed Image:", compressed_image)`
					
					`decompressed_image =`
					`rle_decompress_image(compressed_image)`
					`print("Decompressed Image:")`
					`for row in decompressed_image:`
					`  print(row)`

	A1: Computer fundamentals	7	Cloud computing	A1.1.9 Describe the different types of services in cloud computing. • Services: software as a service (SaaS), platform as a service (PaaS), infrastructure as a service (IaaS) • The differences between the approaches of SaaS, PaaS, and IaaS in various real-world scenarios, recognizing that	

				different degrees of control and flexibility influence resource management and resource availability	
	A1: Computer fundamentals	8	Binary data	A1.2.1 Describe the principal methods of representing data. ● The representation of integers in binary and hexadecimal ● Conversion of binary and hexadecimal integers to decimal, and vice versa ● Conversion of integers from binary to hexadecimal, and vice versa A1.2.2 Explain how binary is used to store data. ● The fundamentals of binary encoding and the impact on data storage and retrieval ● The mechanisms by which data such as integers, strings, characters, images, audio and video are stored in binary form	Possible activity: Program an ESP32 or similar microcontroller to light up LEDs corresponding to the binary representation of a number. Perhaps a binary counter or binary clock?
	A1: Computer fundamentals	9			
	A1: Computer fundamentals	10	Logic gates, truth tables, logic diagrams	A1.2.3 Describe the purpose and use of logic gates. ● The purpose and use of logic gates ● The functions and applications of logic gates in computer systems ● The role of logic gates in binary computing ● Boolean operators: AND, OR, NOT, NAND, NOR, XOR, XNOR A1.2.4 Construct and analyse truth tables. ● Truth tables to predict the output of simple logic circuits ● Truth tables to determine outputs from inputs for a problem description	

				• Truth tables and their relationship to a Boolean expression, with inputs and outputs • Truth tables derived from logic diagrams to aid the simplification of logical expressions • Karnaugh maps and algebraic simplification to simplify output expressions A.1.2.5 Construct logic diagrams. • Logic diagrams to demonstrate how logic gates are connected and interact in a circuit. • Use of standard gate symbols for AND, OR, NOT, NAND, NOR, XOR and XNOR gates • Inputs processed diagrammatically to produce outputs • Combinations of these gates to perform more complex logical operations • Boolean algebra rules to simplify complex logic diagrams and expressions	
	A1: Computer fundamentals	11			
	A1: Computer fundamentals	12			
	A1: Computer fundamentals	13			
	A1: Computer fundamentals	14	Operating systems	A1.3.1 Describe the role of operating systems. • Operating systems abstract hardware complexities to manage system resources A1.3.2 Describe the functions of an operating system. • Maintaining system integrity while running operating systems' background operations	

				• Memory management, file system, device management, scheduling, security, accounting, graphical user interface (GUI), virtualization, networking A1.3.3 Compare different approaches to scheduling. • Managing the execution of processes by allocating CPU time to optimize system performance • First-come first-served, round robin, multilevel queue scheduling, priority scheduling A1.3.4 Evaluate the use of polling and interrupt handling. • Event frequency, CPU processing overheads, power source (battery or mains), event predictability, controlled latency, security concerns • Real-world scenarios may include keyboard and mouse inputs, network communications, disk input/ output operations, embedded systems, real-time systems. A1.3.5 Explain the role of the operating system in managing multitasking and resource allocation. (HL only) • The challenges of multitasking and resource allocation, including task scheduling, resource contention and deadlock	
	A1: Computer fundamentals	15	Operating systems		
	A1: Computer fundamentals	16	Operating systems		

	A1: Computer fundamentals	17	Control systems (HL)	A1.3.6 Describe the use of the control system components. (HL only) ● The input, process, output, and feedback mechanism (open-loop, closed-loop) ● Controller, sensors, actuators, transducers and control algorithm A1.3.7 Explain the use of control systems in a range of real-world applications. (HL only) ● Examples may include autonomous vehicles, home thermostats, automatic elevator controllers, automatic washing machines, traffic signal control systems, irrigation control systems, home security systems, automatic doors.	
	A1: Computer fundamentals	18	Control systems (HL)		
	A1: Computer fundamentals	19	Translation (HL)	A1.4 Translation (HL only) A1.4.1 Evaluate the translation processes of interpreters and compilers. ● The mechanics and use-cases of each translation approach ● The difference in error detection, translation time, portability and applicability for different translation processes, including just-in-time compilation (JIT) and bytecode interpreters ● Example scenarios where the translation method should be considered must include rapid development and testing, performance-critical applications and cross-platform development.	
	A1: Computer fundamentals	20	Exam practice questions		

	A1: Computer fundamentals	21	Assessment		
	A4: Machine learning	1	Intro to ML	A4.1.1 Describe the types of machine learning and their applications in the real world. ● The different approaches to machine learning algorithms and their unique characteristics ● Deep learning (DL), reinforcement learning (RL), supervised learning, transfer learning (TL), unsupervised learning (UL) ● Real-world applications of machine learning may include market basket analysis, medical imaging diagnostics, natural language processing, object detection and classification, robotics navigation, sentiment analysis.	
	A4: Machine learning	2	ML hardware	A4.1.2 Describe the hardware requirements for various scenarios where machine learning is deployed. ● The hardware configurations for different machine learning scenarios, considering factors such as processing, storage and scalability ● Hardware configurations for machine learning ranging from standard laptops to advanced infrastructure ● Advanced infrastructure must include application-specific integrated circuits (ASICs), edge devices, field-programmable gate arrays (FPGAs), GPUs, tensor processing units (TPUs), cloud-based platforms, high-performance computing (HPC) centres.	
	A4: Machine learning	3	Pre-processing (HL)	A4.2.1 Describe the significance of data cleaning. ● The impact of data quality on model performance	

				• Techniques for handling outliers, removing or consolidating duplicate data, identifying incorrect data, filtering irrelevant data, transforming improperly formatted data, and imputation, deletion or predictive modelling for missing data • Normalization and standardization as crucial preprocessing steps A4.2.2 Describe the role of feature selection. • Feature selection to identify and retain the most informative attributes of the data set • Feature selection strategies: filter methods, wrapper methods, embedded methods A4.2.3 Describe the importance of dimensionality reduction. • The curse of dimensionality considerations may include overfitting, computational complexity, data sparsity, the effectiveness of distance metrics, data visualization, sample size increases, memory usage. • Dimensionality reduction of variables, while preserving the relevant aspects of the data. Note: Statistical techniques such as principal component analysis (PCA) and linear discriminant analysis (LDA) are beyond the scope of this course.	
	A4: Machine learning	4	Linear regression (HL)	A4.3.1 Explain how linear regression is used to predict continuous outcomes. • The relationship between the independent (predictor) and dependent (response) variables • The significance of the slope and intercept in the regression equation • How well the model fits the data—often assessed using measures like r2.	

	A4: Machine learning	5	Classification (HL)	A4.3.2 Explain how classifications techniques in supervised learning are used to predict discrete categorical outcomes. ● K-Nearest Neighbours (K-NN) and decision trees algorithms to categorize new data points, based on patterns learned from existing labelled data ● Real-world applications of K-NN may include collaborative filtering recommendation systems. ● Real-world applications of decision trees may include medical diagnosis based on a patient's symptoms.	
	A4: Machine learning	6	Hyper parameters (HL)	A4.3.3 Explain the role of hyperparameter tuning when evaluating supervised learning algorithms. ● Accuracy, precision, recall and F1 score as evaluation metrics ● The role of hyperparameter tuning on model performance ● Overfitting and underfitting when training algorithms	
	A4: Machine learning	7	Clustering (HL)	A4.3.4 Describe how clustering techniques in unsupervised learning are used to group data based on similarities in features. ● Clustering techniques in unsupervised learning group data based on feature similarities ● Real-world applications of clustering may include using purchasing data to segment a customer base.	
EXPECTED OCTOBER HALF TERM BREAK					
	A4: Machine learning	8	Association rule (HL)	A4.3.5 Describe how learning techniques using the association rule are used to uncover relations between different attributes in large data sets.	

				• Mining techniques using the association rule and interpretation of the results for a given scenario For example, in crime analysis, the techniques may reveal that areas with high rates of vandalism also often experience incidents of theft, assisting law enforcement in predictive policing and resource allocation.	
	A4: Machine learning	9	Reinforcement learning (HL)	A4.3.6 Describe how an agent learns to make decisions by interacting with its environment in reinforcement learning. • The principle of cumulative reward and the foundational concepts of agent–environment interaction, encompassing actions, states, rewards and policies • The exploration versus exploitation trade-off as a core concept in reinforcement learning	
	A4: Machine learning	10	Genetic algorithms (HL)	A4.3.7 Describe the application of genetic algorithms in various real-world situations. • For example: population, fitness function, selection, crossover, mutation, evaluation, termination • A real-world application of genetic algorithms is seen in optimization problems, such as route planning (e.g. the “travelling salesperson problem”).	
	A4: Machine learning	11	Artificial neural networks (HL)	A4.3.8 Outline the structure and function of ANNs and how multi-layer networks are used to model complex patterns in data sets. • An artificial neural network (ANN) to simulate interconnected nodes or “neurons” to process and learn from input	

				data, enabling tasks such as classification, regression and pattern recognition ● Sketch of a single perceptron, highlighting its input, weights, bias, activation function and output ● Sketch of a multi-layer perceptron (MLP) encompassing the input layer, one or more hidden layers and the output layer.	
	A4: Machine learning	12	Artificial neural networks (HL)		
	A4: Machine learning	13	Convolutional neural networks (HL)	A4.3.9 Describe how CNNs are designed to adaptively learn spatial hierarchies of features in images. ● Convolutional neural network (CNN) basic architecture: input layer, convolutional layers, activation functions, pooling layers, fully connected layers, output layer ● The effect of the number of layers, kernel size and stride, activation function selection, and the loss function on how CNNs process input data and classify images	
	A4: Machine learning	14	Ethics	A4.4.1 Discuss the ethical implications of machine learning in real-world scenarios. ● Ethical issues may include accountability, algorithmic fairness, bias, consent, environmental impact, privacy, security, societal impact, transparency. ● The challenges posed by biases in training data ● The ethics of using machine learning in online communication may include concerns about misinformation, bias, online harassment, anonymity, privacy.	

				A4.4.2 Discuss ethical aspects of the increasing integration of computer technologies into daily life. ● The importance of continually reassessing ethical guidelines as technology advances ● The potential implications of emerging technologies such as quantum computing, augmented reality, virtual reality and the pervasive use of AI on society, individual rights, privacy and equity	
	A4: Machine learning	15	Ethics		
	A4: Machine learning	16	Model selection	A4.3.10 Explain the importance of model selection and comparison in machine learning. ● How different algorithms can yield different results depending on the data and type of problem ● The reasons for selecting specific machine learning models over others, considering factors like the nature of the problem, its complexity and desired outcomes ● The variability in algorithm performance based on the data's characteristics	
	A4: Machine learning	17	Exam style questions		
	A4: Machine learning	18	Assessment		
	A2: Networks	1	Network types	A2.1.1 Describe the purpose and characteristics of networks. ● Networks: local area network (LAN), wide area network (WAN), personal area network (PAN), virtual private network (VPN)	

				A2.1.2 Describe the purpose, benefits and limitations of modern digital infrastructures. ● Modern digital infrastructure: the internet, cloud computing, distributed systems, edge computing, mobile networks ● Examples where specific networks are used may include the worldwide web (WWW), cryptocurrency blockchains, smart traffic lights, a school.	
	A2: Networks	2	Network devices & transmission media	A2.1.3 Describe the function of network devices. ● Gateways, hardware firewalls, modems, network interface cards, routers, switches, wireless access points ● How devices map to the layers of the TCP/IP model A2.3.2 Compare types of media for data transmission. ● Wired transmission via fibre optic cables and twisted pair cables; wireless transmission ● The advantages and disadvantages of these three types of data transmission ● Factors to consider must include bandwidth, complexity of installation, cost, range, susceptibility to interference, attenuation, reliability, security.	
	A2: Networks	3	Topologies, models, segmentation	A2.2.1 Describe the functions and practical applications of network topologies. ● Network topologies: star, mesh, hybrid ● Factors to consider must include reliability, transmission speed, scalability, data collisions, cost. ● Examples may include home and small office settings, where reliability is paramount, and the use of networks in	

				larger settings (e.g. corporations, government departments, college campuses). A2.2.3 Compare and contrast networking models. ● Client-server and peer-to-peer models ● The respective benefits and drawbacks of client-server and peer-to-peer models ● Real-world applications may include web browsing, email services, online banking, file sharing, VoIP services, blockchain. A2.2.4 Explain the concepts and applications of network segmentation. ● Segmentation for network performance and security, to reduce congestion, to manage network resources efficiently ● Network segmentation must include the uses and roles of segmenting, subnetting and virtual local area networks (VLANs).	
	A2: Networks	4	Protocols	A2.1.4 Describe the network protocols used for transport and application. ● Protocols: transmission control protocol (TCP), user datagram protocol (UDP), hypertext transfer protocol (HTTP), hypertext transfer protocol secure (HTTPS), dynamic host configuration protocol (DHCP)	
	A2: Networks	5	TCP/IP model	A2.1.5 Describe the function of the TCP/IP model. (HL only) ● Application, transport, internet, network interface ● The role of each layer and the interaction between these layers to ensure reliable data transmission over a network	

	A2: Networks	6	Servers	A2.2.2 Describe the function of servers. (HL only) ● Types of servers: domain name server (DNS), dynamic host configuration protocol (DHCP), file server, mail server, proxy server, web server ● Factors to consider must include function, scalability, reliability and security.	
	A2: Networks	7	Practical: Create a socket server		
	A2: Networks	8	IP addressing	A2.3.1 Describe different types of IP addressing. ● The distinction between IPv4 and IPv6 addressing ● The differences between public IP addresses and private IP addresses, and between static IP addresses and dynamic IP addresses ● The role of network address translation (NAT) to minimize the use of IP addresses and to facilitate communication between private internal networks and the public internet	
	A2: Networks	9	Packet switching & routing	A2.3.3 Explain how packet switching is used to send data across a network. ● The process of segmenting data into packets with a routing header attached, and independently transmitting control information, allowing the data to be reassembled at the destination ● The role that switches and routers play in packet switching A2.3.4 Explain how static routing and dynamic routing move data across local area networks. (HL only) ● The process of static routing, and its	

				advantages and disadvantages ● The process of dynamic routing, and its advantages and disadvantages (explanation of a specific routing protocol is not required) ● Factors to consider must include configuration, maintenance, complexity, resource usage, convergence, scalability, network size.	
	A2: Networks	10	Firewalls in detail	A2.4.1 Discuss the effectiveness of firewalls at protecting a network. ● The function of firewalls in inspecting and filtering incoming and outgoing traffic based on whitelists, blacklists and rules ● The strengths and limitations of firewalls ● The role of NAT to enhance network security	
	A2: Networks	11	Vulnerabilities & countermeasures (HL)	A2.4.2 Describe common network vulnerabilities. (HL only) ● Distributed denial of service (DDoS), insecure network protocols, malware, man-in-the-middle (MitM) attacks, phishing attacks, SQL injection, cross-site scripting (XSS), unpatched software, weak authentication, zero-day exploits A2.4.3 Describe common network countermeasures. (HL only) ● Content security policies, complex password policies, DDoS mitigation tools, email filtering solutions, encrypted protocols, input validation (filtering, whitelisting), intrusion detection systems (IDS), intrusion prevention systems (IPS), multifactor authentication (MFA), secure socket layer (SSL) certificate, transport layer security (TLS) certificate, update	

				software, VPNs • The importance of regular security testing and employee training • Wireless security measures may include media access controllers (MAC), whitelists and blacklists.	
	A2: Networks	12			
	A2: Networks	13	Encryption & certificates	A2.4.4 Describe the process of encryption and digital certificates. • The difference between symmetric and asymmetric cryptography • The role of digital certificates in establishing secure network connections • The use of public and private keys in asymmetric cryptography • The significance of encryption key management	
	A2: Networks	14	Practical: TBD		
	A2: Networks	15	Exam style questions		
	A2: Networks	16	Assessment		
EXPECTED WINTER BREAK					
YEAR 13 MOCK EXAMS					
	Case study	1	Introduce the case study	Read the case study. Highlight and identify key points.	
	Case study	2	Understand the text of the case study	Prepare definitions for the terminology list provided at the end of the case study.	
	Case study	3	Understand the text of the case study	Quiz and test yourself and your peers on terminology definitions and introductory concepts.	
	Case study	4	Understand the technology in the case study	Research	

	Case study	5	Understand the technology in the case study	Research	
	Case study	6	Understand the technology in the case study	Present	
	Case study	7	Consider the challenges of the case study	* Research any background information on the challenge * Research and identify real-world examples of the challenges	
	Case study	8	Consider the challenges of the case study	* Research any background information on the challenge * Research and identify real-world examples of the challenges	
	Case study	9	Review for exam questions	Challenge 1	
	Case study	10	Review for exam questions	Challenge 2	
	Case study	11	Review for exam questions	Challenge 3 (HL)	
	Case study	12	Review for exam questions	Challenge 4 (HL)	
	Case study	13	Exam practice questions	Challenge 1	
	Case study	14	Exam practice questions	Challenge 2	
	Case study	15	Exam practice questions	Challenge 3 (HL)	
	Case study	16	Exam practice questions	Challenge 4 (HL)	
END OF COURSE					