Subject	Computer Science	
Title/Topic	**Format**	**Length**
Paper 1 – Computer Systems	Written	2.5 hrs
Paper 2 – Algorithms and Problem Solving	Written	2.5hr

In this Advent assessment I will be asked to show I can…

Paper 1 Content

◆ 1.1.1 Structure and function of the processor

- Purpose of data bus, address bus, control bus
- Comparison of Von Neumann vs Harvard architecture
- Benefits of pipelining
- Benefits of multicore CPUs
- Comparison of RISC vs CISC architectures

◆ 1.1.2 Types of processor

- Differences between CPU and GPU
- Benefits of using a GPU for graphics processing
- RISC vs CISC architectures

◆ 1.1.3 Input, output and storage

- Identifying input/output devices and their uses
- Suitable secondary storage types and justification
- Type of OS needed for multitasking
- Purpose of device drivers

◆ 1.2.1 Systems Software

- Types of operating systems (multitasking)
- How multi-level feedback queue scheduling works
- Role of device drivers

- Purpose of utility software

◆ 1.2.2 Applications Generation

- Benefits of software libraries
- Stages of compilation (lexical analysis, syntax analysis, code generation, optimisation)
- Benefits of using an interpreter
- Benefits of using a compiler

◆ 1.2.3 Software Development

- Relevant to project lifecycle choices in extended-response questions

◆ 1.2.4 Types of Programming Language

- LMC instruction set commands (LDA, BRA, STA, BRZ, BRP)
- Writing simple LMC programs
- **Long-Answer Topic:** Discussing high-level vs low-level language suitability for different projects

◆ 1.3.1 Compression, Encryption, Hashing

- Run-length encoding (decompression)

◆ 1.3.2 Databases

- SQL INSERT statement
- SQL nested SELECT query
- Limitations of flat-file databases
- Entity Relationship Diagrams (ERDs)
- Normalisation (2NF and 3NF requirements)

◆ **1.3.3 Networks**

- **Long-Answer Topic:** Thin client vs thick client approaches, virtual machines, and virtual storage

◆ **1.3.4 Web Technologies**

- JavaScript client-side processing (function completion)

◆ **1.4.1 Data Types**

- Convert denary to 8-bit binary
- Convert denary to hexadecimal
- Convert hexadecimal to denary
- Effect of binary right shift
- Floating-point binary subtraction (normalised form)
- Bitwise AND operation and purpose of bitwise masking

◆ **1.4.2 Data Structures**

- Not directly examined in this paper

◆ **1.4.3 Boolean Algebra**

- Truth table for half adder
- Difference between half adder and full adder
- Logic circuit for full adder
- Simplify Boolean expression using Karnaugh map

◆ **1.5.1 Computing Related Legislation & 1.5.2 Moral and Ethical Issues**

- **Long-Answer Topic:** Legal and moral implications of AI training using internet images

Paper 2 Content

- ◆ **2.1 Elements of Computational Thinking**
  - **2.1.1 Thinking abstractly** – modelling problems, abstraction vs reality
  - **2.1.2 Thinking ahead** – inputs/outputs, preconditions, caching, reusable components
  - **2.1.3 Thinking procedurally** – breaking down problems, identifying sub-procedures
  - **2.1.4 Thinking logically** – decision points, logical conditions, program flow
  - **2.1.5 Thinking concurrently** – identifying parallel tasks, benefits/trade-offs

- ◆ **Sorting Algorithms (2.1.3, 2.1.4 & programming practice)**
  - **Bubble sort** – steps, efficiency improvements (early termination)
  - **Insertion sort** – step-by-step process
  - **Quick sort** – partition function, recursion, divide-and-conquer
  - **Divide-and-conquer** – meaning and examples

- ◆ **Searching Algorithms (2.1.2, 2.1.4)**
  - **Linear vs binary search** – features, preconditions, big-O complexity
  - **Long-Answer Topic:** Comparing linear vs binary search in a real-world scenario (specification link: **2.1.2, 2.1.4**)
  - **Binary search trees** – balanced vs unbalanced, traversal methods

- ◆ **Data Structures – Stacks (1.4.2 Data Structures)**
  - **Stack operations** – push, pop, pointer management
  - **Implementing a stack** using an array (pseudocode)

- **Applications of stacks** – e.g., reversing data

◆ **Data Structures – Graphs (1.4.2 Data Structures)**

- **Graph terminology** – nodes, edges, directed/undirected, weighted
- **Dijkstra's algorithm** – finding shortest path in a weighted graph
- _A algorithm*_ – use of heuristics for efficiency
- **Graph as a visualisation** – representing real-world problems

◆ **Data Structures – Linked Lists (1.4.2 Data Structures)**

- **Features of linked lists** – dynamic size, nodes, pointers
- **Object-oriented implementation** – classes for node and linked list
- **Traversal and manipulation** – inserting, removing, outputting nodes
- **Long-Answer Topic:** Comparing linked list implementations (OOP vs 2D array)

◆ **Data Structures – Trees (1.4.2 Data Structures)**

- **Binary search trees** – insertion, balancing, traversal (post-order, breadth-first)
- **Balanced vs unbalanced trees** – impact on search efficiency

◆ **Data Structures – Arrays & 2D Arrays (1.4.2 Data Structures)**

- **Using arrays to store frequency data** – error tracing in pseudocode
- **2D arrays for linked list simulation** – alternative to OOP

◆ **Programming Concepts (1.2.4 Types of Programming Language)**

- **Global vs local variables** – differences, advantages, drawbacks

- **Parameter passing** – by value vs by reference
- **IDE debugging features** – e.g., breakpoints, variable inspection, stepping

◆ **Recursion & Iteration (2.1.3, 2.1.4)**

- **Identifying recursive functions** – e.g., quickSort
- **Types of iteration** – e.g., for loops in partition function

◆ **Computational Methods & Thinking (2.1 Elements of Computational Thinking)**

- **Divide-and-conquer** – used in quick sort
- **Data mining** – meaning and applications in business
- **Other computational methods** – e.g., backtracking, greedy algorithms, dynamic programming

◆ **Algorithm Efficiency & Big-O Notation (2.1.2)**

- **Understanding time/space complexity** – O(n), O(log n), O(1)
- **Applying complexity analysis** to real problems (e.g., searching large datasets)

◆ **Processor Design & Pipelining (1.1.1 Structure and function of the processor)**

- **Long-Answer Topic:** Pipelining in processors – stages, efficiency, managing multiple instructions
- **Visualising instruction processing** – fetch, decode, execute overlap

What should I do to revise and prepare for this assessment?

What useful websites/resources could I use to help me prepare?