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

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

Paper 1

Databases

- Explain the limitations of flat-file databases.
- Draw Entity Relationship Diagrams (ERDs).
- Understand primary keys and foreign keys.
- Explain the requirements of Second Normal Form (2NF).
- Explain the requirements of Third Normal Form (3NF).
- Write SQL INSERT statements.
- Write SQL queries using nested SELECT statements.

CPU and GPU

- Compare the roles of a CPU and GPU.
- Explain the advantages of using a GPU for graphics processing.

Programming

- Complete partially written algorithms or code.
- Use arrays effectively.
- Use loops and selection.
- Use parameters and return values in functions.
- Access and manipulate HTML elements (JavaScript/pseudocode).

Number Systems

- Convert between denary and binary.
- Convert between denary and hexadecimal.
- Convert between hexadecimal and denary.

Software Development

- Explain the benefits of using software libraries.

Operating Systems

- Identify different types of operating systems (e.g. multitasking).

- Explain how a Multi-Level Feedback Queue (MLFQ) scheduler works.
- Explain why hardware devices require device drivers.
- Identify common utility software and explain its purpose.

Compression

- Decompress data using Run Length Encoding (RLE).

Programming Languages

- Explain the differences between high-level and low-level programming languages.
- Identify suitable applications for high-level and low-level languages.
- Justify the choice of programming language for different scenarios.

Input, Output and Storage

- Select appropriate input devices for different scenarios and justify your choice.
- Select appropriate output devices for different scenarios and justify your choice.
- Choose suitable secondary storage devices and explain why they are appropriate.

Binary Arithmetic and Data Representation

- Explain the effect of left and right binary shifts.
- Perform floating-point binary calculations, including normalisation.
- Apply bitwise AND masks.
- Explain the purpose of bit masking.

Ethical, Legal, Moral and Cultural Issues

- Explain what Artificial Intelligence (AI) is.
- Discuss the legal implications of AI.
- Discuss the moral and ethical implications of AI.
- Evaluate different viewpoints and produce a justified conclusion.

Computer Architecture

- Explain the purpose of the data bus, address bus and control bus.
- Compare the Von Neumann and Harvard architectures.
- Explain the benefits of pipelining.
- Compare RISC and CISC processor architectures.
- Explain the advantages of multicore processors.

Translators

- Describe the stages of compilation:
    - Lexical analysis

    - Syntax analysis
    - Code optimisation
    - Code generation
- Explain the advantages of interpreters.
- Explain the advantages of compilers.

Virtualisation and Cloud Computing

- Explain what is meant by virtual machines.
- Explain what is meant by virtual storage.

Assembly Language (Little Man Computer)

- Recognise LMC instructions and their functions.
- Write simple LMC programs using input, output, branching and memory locations.
- Trace and debug simple LMC programs.

Paper 2

Variables and Scope

- Explain the difference between global and local variables.
- Describe the advantages and disadvantages of global variables.
- Describe the advantages and disadvantages of local variables.

Graphs and Graph Traversal

- Explain how graphs model real-world problems.
- Apply Dijkstra's shortest path algorithm.
- Complete Dijkstra's working table.
- Identify the shortest path and total cost.
- Explain how the A* algorithm uses heuristics.
- Compare heuristic searching with exhaustive searching.

Testing and Debugging

- Identify logical, syntax and runtime errors.
- Debug pseudocode by identifying and correcting errors.
- Trace code to locate mistakes.

Searching Algorithms

- Compare linear search and binary search.
- Explain the preconditions required for binary search.
- Compare the efficiency of searching algorithms using Big O notation.
- Justify the most appropriate search algorithm for a given scenario.

Linked Lists

- Explain the features of a linked list.
- Compare linked lists with arrays.
- Traverse a linked list.
- Insert new nodes into a linked list.
- Remove nodes from a linked list.
- Explain how pointers (references) connect nodes.

Big O Complexity

- Interpret Big O notation.
- Compare the time and space complexity of algorithms.
- Use Big O notation when evaluating algorithms.

Processor Architecture

- Explain how pipelining works.
- Show how instructions move through a processor pipeline.
- Explain the advantages and disadvantages of pipelining.
- Evaluate whether pipelining is suitable for a given scenario.

Parameter Passing

- Explain parameter passing by value.
- Compare pass-by-value and pass-by-reference.

Object-Oriented Programming (OOP)

- Understand classes, objects and attributes.
- Write constructor methods.
- Write getter methods.
- Write setter methods.
- Create objects from classes.
- Call object methods.
- Understand encapsulation through private attributes.

Algorithms and Programming

- Write complete algorithms using pseudocode or program code.
- Trace algorithms to determine outputs.
- Complete partially written algorithms.
- Identify and correct errors in algorithms.

Computational Thinking

- Explain divide-and-conquer.
- Identify other computational thinking methods, including:
    - Abstraction
    - Decomposition

- Pattern recognition
- Algorithmic thinking

Stacks

- Show the effect of PUSH and POP operations.
- Trace the contents of a stack after operations.
- Write a PUSH function.
- Write a POP function.
- Use a stack to solve problems such as reversing data.

Integrated Development Environments (IDEs)

- Identify common debugging features of an IDE.
- Explain how breakpoints are used.
- Explain stepping through code.
- Explain variable watches.
- Explain syntax highlighting and error reporting.

Sorting Algorithms

- Describe how a bubble sort works.
- Explain optimisations that improve the efficiency of a bubble sort (e.g. early exit if no swaps occur).
- Describe how an insertion sort works.
- Trace and explain how a quick sort works.
- Identify recursive functions within an algorithm.
- Identify different types of iteration used in algorithms.
- Explain the purpose of the partition function in quick sort.
- Modify a sorting algorithm to sort in ascending or descending order.
- Explain the divide-and-conquer approach used by quick sort.

Extended Response (9–12 Mark Questions)

Students should be able to:

- Compare algorithms using Big O notation.
- Evaluate search algorithms for different scenarios.

What should I do to revise and prepare for this assessment?
• Go back through notes/handouts • Watch the relevant CraignDave videos • Complete practice papers given out • Complete Seneca activites (if appropriate)

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