Unit 1 RAG Sheet

# A - Computational Thinking

|  |  |  |  |
| --- | --- | --- | --- |
| **Decomposition** |  |  | X |
| Identifying and breaking down problems and processes |  |  | X |
| Breaking down problems into distinct steps |  |  | X |
| Describing problems and processes as a set of structured steps |  |  | X |
| Communicating the key features of problems and processes to others |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Pattern Recognition** |  |  | X |
| Identifying common elements or features in problems or systems |  |  | X |
| Identifying and interpreting common differences between processes or problems |  |  | X |
| Identifying individual elements within problems |  |  | X |
| Describing patterns that have been identified |  |  | X |
| Making predictions based on identified patterns |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Pattern Generalisation and Abstraction** |  |  | X |
| Identifying information that is necessary to solve an identified problem |  |  | X |
| Filtering out information that is not needed to solve an identified problem |  |  | X |
| Representing parts of a problem or system in general terms by identifying the following: |  |  | X |
| Variables |  |  | X |
| Constants |  |  | X |
| Key processes |  |  | X |
| Repeated processes |  |  | X |
| Inputs |  |  | X |
| Outputs |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Algorithm Design** |  |  | X |
| Describing a step by step strategy to solve a problem |  |  | X |

# B - Standard Methods and Techniques to Design Algorithms

|  |  |  |  |
| --- | --- | --- | --- |
| **Pseudocode - Interpretation** |  |  | X |
| Interpreting pseudocode using the following: |  |  | X |
| Applying processes to calculate outcomes |  |  | X |
| Evaluating the structure and logic of given code against given requirements |  |  | X |
| Suggest improvements to logical structures and processes |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Pseudocode - Anaylsis** |  |  | X |
| Improvement the effectiveness and efficiency of code |  |  | X |
| Identify and fix errors within code |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Pseudocode - Creation** |  |  | X |
| Sequence |  |  | X |
| Structure including hierarchy and indentation |  |  | X |
| Operations including the following: |  |  | X |
| BEGIN |  |  | X |
| END |  |  | X |
| INPUT |  |  | X |
| OUTPUT |  |  | X |
| PRINT |  |  | X |
| READ |  |  | X |
| WRITE |  |  | X |
| Decisions including the following: |  |  | X |
| IF |  |  | X |
| THEN |  |  | X |
| ELSE |  |  | X |
| ELSE IF (ELIF) |  |  | X |
| WHEN |  |  | X |
| Repetition including the following: |  | X |  |
| FOR |  | X |  |
| REPEAT UNTIL |  |  | X |
| WHILE |  | X |  |
| WHILE NOT |  | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Flowcharts** |  |  | X |
| Recognition and use of the following symbols: |  |  | X |
| Process |  |  | X |
| Decisions |  |  | X |
| Input / Output |  |  | X |
| Connectors |  |  | X |
| Start / End |  |  | X |

# C – Programming Paradigms

|  |  |  |  |
| --- | --- | --- | --- |
| **Handling Data within a Program** |  |  | X |
| Defining and declaring data structures, variables and constants of the following types: |  |  | X |
| Alphanumeric Strings |  |  | X |
| Arrays |  |  | X |
| Boolean |  |  | X |
| Characters |  |  | X |
| Date/Time |  |  | X |
| Floating Point |  |  | X |
| Integers |  |  | X |
| Objects |  |  | X |
| Records |  | X |  |
| Set |  | X |  |
| Strings |  |  | X |
| Managing the above according to: |  |  | X |
| Global / local variable scope |  |  | X |
| Naming conventions |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Arithmetic Operations** |  |  | X |
| Identify and be able to use the following operators: |  |  | X |
| + |  |  | X |
| - |  |  | X |
| / (DIV) |  |  | X |
| \* |  |  | X |
| % (MOD or modulo) |  |  | X |
| Relational operators (==, <, >, <>, <=, >=) |  |  | X |
| Boolean operators (AND, OR, NOT) |  |  | X |
| Date/Time |  | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Built-in Functions** |  |  | X |
| Arithmetic functions including: |  |  | X |
| Random |  |  | X |
| Range |  |  | X |
| Round |  |  | X |
| Truncation |  | X |  |
| String handling functions including: |  |  | X |
| Concatenation |  | X |  |
| Length |  |  | X |
| Position |  |  | X |
| String conversion (e.g. int to string, string to int) |  |  | X |
| General Functions: |  |  | X |
| Input |  |  | X |
| Open |  |  | X |
| Print |  |  | X |
| Range |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Validating Data** |  |  | X |
| Validation check techniques such as: |  |  | X |
| Data type |  |  | X |
| Range |  |  | X |
| Constraints |  | X |  |
| Boolean |  |  | X |
| Post-check actions |  | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Control Structures** |  |  | X |
| Loops including: |  |  | X |
| REPEAT |  |  | X |
| FOR |  |  | X |
| WHILE |  |  | X |
| BREAK |  |  | X |
| Branches: |  |  | X |
| IF |  |  | X |
| THEN |  |  | X |
| ELSE |  |  | X |
| ELSE IF (ELIF) |  |  | X |
| Function Calls: |  |  | X |
| Defining functions |  |  | X |
| Declaring arguments |  |  | X |
| Calling functions |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Structures** |  | X |  |
| Lists |  |  | X |
| Single dimensional arrays |  |  | X |
| Multi-dimensional arrays |  | X |  |
| Records |  | X |  |
| Sets |  | X |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Common and Standard Algorithms** |  |  | X |
| Sorting: |  |  | X |
| Bubble sort |  |  | X |
| Quick sort |  |  | X |
| Insertion sort |  |  | X |
| Searching: |  |  | X |
| Serial / Linear search |  |  | X |
| Binary search |  |  | X |
| Other standard algorithms: |  | X |  |
| Count occurrences |  | X |  |
| Input validation |  |  | X |
| Using stacks and queues to implement sorting and searching: |  |  | X |
| Last in First out |  |  | X |
| First in First out |  |  | X |

# D - Types of programming and mark-up languages

|  |  |  |  |
| --- | --- | --- | --- |
| **Procedural Programming** |  |  | X |
| Structure: |  |  | X |
| Statements |  |  | X |
| Blocks |  |  | X |
| Procedures |  |  | X |
| Functions / Sub-routines |  |  | X |
| Control Structures: |  |  | X |
| Sequence |  |  | X |
| Conditional |  |  | X |
| Iterative |  |  | X |
| Interpret, debug and use code written using procedural paradigms |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Oriented Programming** |  |  | X |
| Structure: |  |  | X |
| Classes |  |  | X |
| Objects / Instances |  |  | X |
| Features: |  |  | X |
| Inheritance |  |  | X |
| Encapsulation |  | X |  |
| Polymorphism and overloading |  | X |  |
| Data hiding |  | X |  |
| Reusability |  |  | X |
| Interpret, debug and use code written using object oriented paradigms |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Event Driven Programming** |  |  | X |
| Structure: |  |  | X |
| Main loop |  |  | X |
| Callback functions |  |  | X |
| Sub-routines |  |  | X |
| Features: |  |  | X |
| Events |  |  | X |
| Event handlers |  |  | X |
| Event loops |  |  | X |
| Service orientated processing |  |  | X |
| Time driven |  | X |  |
| Trigger functions |  |  | X |
| Interpret, debug and use code written use event driven paradigms |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Coding for the web** |  |  | X |
| Characteristics, features and implications of mark-up and web languages in relation to: |  |  | X |
| Performance |  |  | X |
| Platform independence |  |  | X |
| Power |  | X |  |
| Protocols |  |  | X |
| Security |  |  | X |
| Uses, applications and implications of mark-up web languages |  |  | X |
| Interpret, debug and use HTML5 |  |  | X |
| Uses, applications and implications of client-side scripting |  |  | X |
| Uses, applications and implications of server-side scripting |  | X |  |
| Issues and implications of implementing code on a web platform |  |  | X |

|  |  |  |  |
| --- | --- | --- | --- |
| **Translation** |  |  | X |
| Reasons for translating code from one language to the other |  |  | X |
| Benefits of translating code from one language to the other |  |  | X |
| Drawbacks of translating code from one language to the other |  |  | X |
| Implications of translating code in particular on: |  |  | X |
| Users |  |  | X |
| Organisations |  |  | X |
| Developers |  |  | X |
| Alternative ways to implement current code base |  |  | X |