Computational Thinking Techniques

Ref.	Category	Activity
AL1	Algorithms	Writing instructions that if followed in a given order (sequences) achieve a desired effect
AL2	Algorithms	Writing instructions that use arithmetic and logical operations to achieve a desired effect
AL3	Algorithms	Writing instructions that store, move and manipulate data to achieve a desired effect; (variables and assignment)
AL4	Algorithms	Writing instructions that choose between different constituent instructions (selection) to achieve a desired effect;
AL5	Algorithms	Writing instructions that repeat groups of constituent instructions (loops/iteration) to achieve a desired effect;
AL6	Algorithms	Grouping and naming a collection of instructions that do a well-defined task to make a new instruction (subroutines, procedures, functions, methods);
AL7	Algorithms	Writing instructions that involve subroutines use copies of themselves to achieve a desired effect (recursion);
AL8	Algorithms	Writing sets of instructions that can be followed at the same time by different agents (computers or people) to achieve a desired effect (Parallel thinking and processing, concurrency);
AL9	Algorithms	Writing a set of rules to achieve a desired effect (declarative languages);
AL10	Algorithms	Using a standard notation to represent each of the above;
AL11	Algorithms	Creating algorithms to test a hypothesis;
AL12	Algorithms	Creating algorithms that give good, though not always the best, solutions (heuristics);
AL13	Algorithms	Creating algorithmic descriptions of real world processes so as to better understand them (computational modelling);
AL14	Algorithms	Designing algorithmic solutions that take into account the abilities, limitations and desires of the people who will use them;
EV1	Evaluation	Assessing that an algorithm is fit for purpose;
EV2	Evaluation	Assessing whether an algorithm does the right thing (functional correctness);
EV3	Evaluation	Designing and running test plans and interpreting the results (testing);
EV4	Evaluation	Assessment whether the performance of an algorithm is good enough;
EV5	Evaluation	Comparing the performance of algorithms that do the same thing;
EV6	Evaluation	Making trade-offs between conflicting demands;
EV7	Evaluation	Assessment of whether a system is easy for people to use (usability);

EV8	Evaluation	Assessment of whether a system gives an appropriately positive experience when used (user experience);
EV9	Evaluation	Assessment of any of the above against set criteria;
EV10	Evaluation	Stepping through algorithms/code step by step to work out what they do (dry run / tracing);
EV11	Evaluation	Using rigorous argument to justify that an algorithm works (proof);
EV12	Evaluation	Using rigorous argument to check the usability or performance of an algorithm (analytical evaluation);
EV13	Evaluation	Using methods involving observing a system in use to assess its usability or performance (empirical evaluation);
EV14	Evaluation	Judging when an algorithmic solution is good enough even if it is not perfect;
EV15	Evaluation	Assessing whether a solution meets the specification (criteria);
EV16	Evaluation	Assessing whether a product meets general performance criteria (heuristics)
DE1	Decomposition	Breaking down artefacts (whether objects, problems, processes, solutions, systems or abstractions) into constituent parts to make them easier to work with
DE2	Decomposition	Breaking down a problem into simpler but otherwise identical versions of the same problem that can be solved in the same way (Recursive and Divide and conquer strategies)
AB1	Abstraction	Reducing complexity by removing unnecessary detail;
AB2	Abstraction	Choosing a way to represent artefacts (whether objects, problems, processes or systems) to allow it to be manipulated in useful ways;
AB3	Abstraction	Hiding the full complexity of an artefact, whether objects, problems, processes, solutions, systems (hiding functional complexity);
AB4	Abstraction	Hiding complexity in data, for example by using data structures;
AB5	Abstraction	Identifying relationships between abstractions;
AB6	Abstraction	Filtering information when developing solutions;
GE1	Generalisation	Identifying patterns and commonalities in problems, processes, solutions, or data.
GE2	Generalisation	Adapting solutions or parts of solutions so they apply to a whole class of similar problems;
GE3	Generalisation	Transferring ideas and solutions from one problem area to another