Confident with complexity

I can discover the most important part of a problem

I can break complex problems into parts

I can concentrate on the most important part

I can explain how I used decomposition & abstraction to cope with complex problems

Iterative experimenter & debugger

I repeatedly experiment through making, testing & debugging

I can develop, debug and test more than once until a product is refined

I can explain how the iteractyive cycle improves my work

Persistent problem solver

I can persevere when solving difficult problems even if the solution is not obvious

I learn from setbacks and don't let them put me off from solving difficult problems

I can describe how I overcame problems to arrive at a solution

Evaluator

I can evaluate my solutions against set criteria

I can design criteria to evaluate my creations

I can explain how evaluation helped me improve my project

Tolerant of ambiguity

I recognise that there is often more than one way to solve a problem

I recognise that there is often more than one way to describe a problem

I can explain how I managed ambiguity in a project

Adapter of solution to solve new problems

I can identify patterns in problems and solutions

I can adapt existing ideas to solve new problems

I can explain how I adapted a solution to solve a different problem

Communicator & team builder

I can contribute useful ideas to a partner or group

I can encourage others to share gtheir ideas

I can lead by using all the people

Confident with open ended problems

I am not happy just accepting the first solution

I look for a range of solution to the same problem

I can describe how a project could be extended

Step 1	Step 2	Step 3
Step 4	Step 5	Pre-requisites for success (prior knowledge) Equipment Debugging Support (cards, methods etc)

Project Name | Year Group

Adapting a programming project to make sure it Includes computational thinking & wide range of problem solving skills

Computational Thinkers

Algorithmic thinkers

Think through steps or rules to achieve something

Algorithmic evaluators

Which algorithm is best, most efficient?

Decomposers

Break problem up into parts and solve parts separately

Abstractors

Find the most important part of problem

Generalisers

Adapt solution from one problem to solve something else

Programming Constructs

Sequence

Repetition

Selection

Variables

Procedure

Function

List / Array

Other programming considerations

Physical Computing

Inputs

Outputs