**Computational Thinking in the Primary Computing Curriculum**

**What is computational thinking?**

Wikipedia2 describes it as “problem solving method that uses computer science techniques”



 .

ITSE1 in their article on computational thinking describe computational thinking as critical thinking ideas combined with the power of computing. They also have a short video to help define it which is worth viewing 1a

**What thinking skills are included?**

Computer scientists and educationalist are in broad agreement as to what should be included. Miles Berry6 included algorithms, decomposition, patterns, logical reasoning and abstraction. A recent CAS working group included algorithmic thinking, evaluation, decomposition, abstraction and generalization in their very helpful framework document7.

**What do these thinking skills mean for primary pupils and teachers?**

**Algorithmic Thinking**

Defining a precise set of instructions or rules to achieve an outcome or solve a problem. A recipe can be an algorithm, musical notation can be an algorithm. Instructional writing can be an algorithm. All working computer programs started life as human ideas that were expressed as algorithms in thoughts, words, symbols or flow charts. Programming is the challenge of turning precise ideas (algorithms) into code that can be read by a machine. When we define a precise set of instructions we save time as this algorithm can be reused to solve a problem over and over again and adapted to solve similar problems.

**Evaluation**

Evaluation is how we look at algorithms and determine how useful they are, how adaptable, how efficient, how correct. There may be many algorithmic solutions to a problem, evaluation asks which one was best and why? Evaluation is also concerned with the people who use an algorithm. Did it solve their problem? Was it better on paper than in practice? Evaluation is also a very useful skill to extend into programming as well. Getting pupils to think about an end user in the design (algorithm) stage can help focus ideas. I think there is a lot of similarity between logical thinking in the national curriculum and evaluation.

**Decomposition**

Decomposition is the skill of breaking a complex problem up into smaller manageable chunks and solving these chunks separately. I have found this to be a wonderfully useful skill in games design. Faced with the task of creating a new game8 pupils are often overwhelmed by the amount to think through. We use a decomposed planner where they jot down what they want the game to do first before circling objects and ideas and describing these in detail. This allows them to focus on designing a small parts of the game separately before recomposing these ideas into the whole.

**Abstraction**

Abstraction is the skill of reducing complexity by hiding irrelevant detail and focussing on the most important element. This is a really useful computational skill as once the irrelevant detail has been stripped away computer scientists can focus on what really needs doing. Imagine I wanted to turn the game matching pairs into a computer game. The most important element is; you win if items are the same. This can be abstracted further into A = B win, A ≠ B lose. We recently used abstraction to turn a musical sound track on a video into an algorithm and then into musical programming in Scratch. We started by listening to a video and identifying all the elements on the video, singing, high and low pitch notes, moving pictures, backing track etc. We then looked at what detail was important to turn into notes on Scratch and what was irrelevant. We ended up only keeping the high and low pitch notes. We swapped to a much simpler music track with notes only and listened to this to write a musical algorithm before converting this into code.9

**Generalization**

Generalization is adapting a solution that solved one problem to solve another. In our abstraction example earlier we reduced matching pairs to A = B win A ≠ B lose. We could then use generalisation to adapt this solution to think about creating a quiz. In a quiz, if the answer we have thought of is the same as the users answer we say it is correct. If the answer is not the same we say it is wrong. This is almost identical to A = B win A ≠ B wrong so we can adapt one solution to solve a similar problem. In our Scratch perimeter program10 we discover a simple way to calculate the diameter of a circle by multiplying the radius by two. Pupils then use the principle of generalization to adapt this solution to calculate the perimeter of regular 2D shapes.