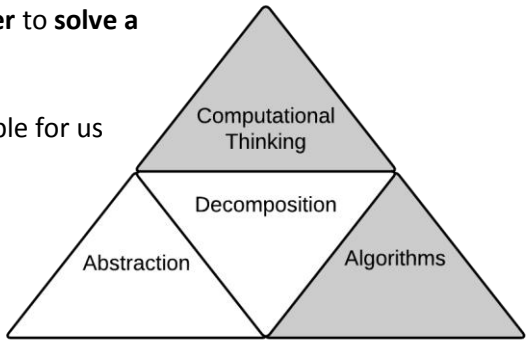
R 1

Algorithms - Reading

An **algorithm** is **a sequence of steps** which are **carried out in order** to **solve a problem**.



If we can write down the steps to solve a problem then it is possible for us to program a computer to carry them out and **automate** them.

Algorithms do not need computers to carry them out. They are simply the steps required to solve a problem. So a recipe gives an algorithm for creating a meal. It is normally carried out by a person, but it could be carried out by a machine in a factory. In mathematics, you are taught algorithms for how to solve problems. These are the steps you need to follow.

The following stages need to be used to create an algorithm:

1. State the problem clearly and **precisely**
2. Break the problem down into smaller problems (**decomposition**)
3. Try to find patterns or similarities to make smaller problems abstract (**abstraction**)
4. Create a step-by-step solution to solve each of these smaller problems (the algorithm)

As algorithms are simply the steps to solve a problem, we normally write them down in **pseudocode** or by using a **flow diagram**. Pseudocode can be written in a similar way to a programming language, but simple statements in English are also okay. The only important point is that pseudocode is written in a way that allows other programmers to understand how the algorithm works.

A **computer program** is an algorithm that has been implemented using a specific **programming language**.

Algorithmic thinking is the ability to be able to solve problems by creating an algorithm. The following are some of the key components of algorithmic thinking which we use when trying to create algorithms.

Component of algorithmic thinking	What this means
Functional decomposition	Splitting up a problem into smaller problems which each have their own functionality. These will be programmed using **subroutines** such as functions or procedures
Repetition	**Iteration** (repeating sections of code) or **recursion** (functions calling themselves)
Selection	The ability to run different parts of the algorithm based on a **condition**
Data structures	These include **arrays**, **lists** and **records**.
Abstraction	Ignores details of a problem to make it easier to solve
Parameters	These are the limits of the inputs that are used in the algorithm. So for instance, an algorithm which adds two numbers together might have parameters that the numbers must be **integers** (whole numbers) and positive.

One objective of algorithmic thinking is to create **efficient** algorithms. If an algorithm is efficient then it will use fewer **computational resources** and complete the problem faster.