**A1 – Decomposition:**

**Problem Decomposition:** (Breaking down a complex problem or system into smaller parts that are more manageable & easier to understand.) / (Break a complicated task into a set of identifiable subtasks.)

* This involves dividing a large & complex problem into smaller sub-problems & splitting these problems further until each problem can be solved. These sub-problems might then be able to be developed as programmable modules (functions/procedures)
* Decomposition allows the smaller sub-problems to be easily solved compared to having one larger and more complex problem. This is because larger problems are daunting. Abstraction can help with this process. The smallest type of sub-problem/task is called ‘atomic’.
* This allows divide & conquer to be used.
* This can also increase the speed of production in programming.
* This can also reduce the amount of active processing & memory requirements.
* This is good in teamwork as programmers can be assigned to areas that match their specialties. However, this can introduce errors if each subprogram/module cannot interact correctly.

**Composition:** (Combing objects or data types into more complex ones.)

* Combining the atomic procedures, which were created from solving smaller-sub problems, to generate a complex structure.
* For example, assemble the modules. Each sub-program will produce a result, which then the output of one module will become the input to the next one.

**A2 – Pattern Recognition:**

**Pattern Recognition**: (the ability to identify recurring characteristics within the same problems and between different problems)

* If a new problem may have features that are similar problems compared to previously solved problems. Then recognising these patterns can make problem solving much easier.

**You Must:**

* **Be able to identify common elements/features in problems.** (Recognising these as patterns)
* **Identifying and interpreting common differences between processes or problems**
* **Identifying individual elements within problems**
* **Describing patterns that have been identified**
* **Making predictions based on identified patterns:**

**A3 – Pattern Generalisation & Abstraction:**

**Abstraction** (brief explanation): – This is a problem solving tool that involves the process of simplifying a situation by separating ideas from particular instances or reality.

**Types of Abstraction:**

**Representational Abstraction:** – This is the process of removing any unnecessary detail from the problem down to its essential features. This can make it easier to find a computational solution.

**Generalisation Abstraction:** – This is the process of simplifying a problem by grouping the remaining features in a hierarchical structure.

*Examples:*

* A Variable.
* A data structure.
* A network address.
* Layers in a network.
* A symbol on a map.

**The Main Advantages:**

* It allows to focus on key points within the problem. This prevents distraction from unimportant details.
* It allows to see the overall structure of the problem.
* It can allow people to work into teams.
* Removes unneeded complexities that may require more programming or computational resources, such as design effort & extra memory.
* It can help to find common patterns within a complex problem.
* Allows the use of generalising from many examples.
* Allows us to simplify structures from the complex system.

**Context is key:** – The context of the problem allows us to see what is relevant to the solution and what is not. Context also allows us to see any similarities in previous problems that we have possibly solved in the past. It is important to note that some details may be considered irrelevant to a computational solution, but significant in human context.

**IMPORTANT:** Abstraction strips unnecessary detail, which are considered irrelevant to the problem, from reality. This can be used to create a model to show only the parts of reality that we need to be able to observe, understand, discuss and use. For example, abstraction may be used in the creation of a flying simulator program to teach new pilots. As a result, details such as birds, buildings on the ground, people may not be used in the program as it’s not related to its purpose. However, weather, clouds and day/night cycle may be kept for the training program. It depends on the context. The message is that Abstraction is not an overall representation of reality. Reality will include these features obviously whereas an abstracted program will not.

**Pattern Generalisation: -** This occurs when relationships between patterns can be identified and simple conclusions can be drawn. For example, patterns can be identified even when, at first, it does not look like there are many similarities.

**Representing parts of a problem or system in general terms:**

* **Variables** are identifiers (name) given to memory locations whose contents will change during the course of the program.
* A **Constant** is a named location that can be used to store value that never changes while the program is running.
* **Key Processes.** These are the processes that are absolutely critical to understanding a problem or how a system works.
* **Repeated Processes.** These are processes that occur multiple times within a problem.
* **Inputs.** These are the values entered into the system.
* **Outputs.** This is information presented to the user in a required format.

**A4 – Algorithm Design:**

**Algorithm:** A set of instructions that is followed in order to solve a problem or to perform a specific stage of processing in the overall solution. Programs may be made from a collection of many different algorithms.

**Algorithmic thinking:**A logical way of getting from the problem to the solution on a step by step strategy. For example, if the steps you take to solve a problem follow an algorithm then they can be reused and adapted to solve similar problems in the future.