Revision Questions for Section 4.1 - Computational thinking, problem-solving and programming - General principles

Thinking procedurally

4.1.1 Links: Subroutines | Black Boxes | Procedures | Top-down Design

a. List the **steps** required to make a cup of tea.
b. Identify the **sequence** of colours in a traffic light.
c. Outline the **steps** required to bake a cake.
d. Think of any example when following an incorrect sequence of steps will **not** result in the required outcome.
e. Explain **two** advantages of the 'top down' approach to problem solving i.e. breaking a larger problem into smaller parts
f. Explain the role of **sub-procedures** (or methods) in solving a problem.

Thinking logically

4.1.4 Links: Decision Making | Conditional Statement | Switch Statement | Rock-Paper-Scissors | Missionaries & Cannibals

a. Identify **when** a decision is required in a temperature control system.
b. Identify the **decisions** required to choose the winner in a game of *rock-paper-scissors.*
c. Identify the **conditions** that help decide which move to make in the *missionaries-and-cannibals* river crossing problem.
d. Explore the **relationship** between IF..THEN..ELSE and the SWITCH statement.

Thinking ahead

4.1.9 Links: Input-Output | Gantt Chart | Preconditions and Postconditions | Preconditions | Exceptions

a. Identify the **inputs** and **outputs** required for baking a cake, calculating bank interest and calculating how old a person is in years, months and days.
b. Outline how a **Gantt chart** can be helpful in planning a project.
c. Suggest possible **pre-planning** that might happen before you make a cake, build a house, or program a new app.
d. Outline the need for **pre-conditions** before running an algorithm.
e. Outline possible **pre-** and **post-conditions** for baking a cake, finding the square root of a number and sorting a list of numbers.
f. Identify possible **exceptions** that should be considered in the preconditions when calculating an end-of-year bonus for all employees, and finding the numeric average of the contents of a list.

Thinking concurrently

4.1.14 Links: Concurrency | Dining Philosophers | Multi-tasking

a. Identify possibly **concurrent** parts of an algorithm for baking a cake, building a house, and performing a binary search.
b. Describe how **concurrent processing** could be used when building a house or performing speech recognition.
c. Evaluate the **Dining Philosophers** problem as a model of concurrency.

Thinking abstractly

4.1.17 Links: Abstraction | Objects | Rock-Paper-Scissors | Black Boxes

a. Identify how the idea of **abstraction** can apply in a car engine, a cake shop, and an iPad.
b. Explain the **need** for abstraction when deriving a computational solution for a school database problem.
c. **Construct** an abstraction for a typical school database, the *rock-paper-scissors* game, and a computer chess game.
d. Distinguish between a **real-world** entity and its **abstraction**, such as a map or a car manual.