Advanced Higher
Computing Science

Software Design and Development

Design Notes

Name:_____

DESIGN

TOP-DOWN DESIGN / STEPWISE REFINEMENT

Top-down design involves identifying an overall problem and breaking it down into smaller sub-problems (main steps).

The process of stepwise refinement is then used to break the sub-problems down until each one is small enough that they are manageable.

Top-down design emphasises planning and a complete understanding of the system.

No coding should take place until a sufficient level of detail has been reached in the design.

Each sub-problem is coded as a module however this delays testing of the functional units until significant design is complete.

DESIGN TECHNIQUES

Pseudocode

When using pseudocode to design efficient solutions to a problem, you must include the following:

- **Top level design** — the major steps of the design. In the example below, numbered from 1 to 4.
- **Data flow** — shows the information that must flow In or Out from the sub-programs. In the example below, written to the right of the top level design.
- **Refinements** — break down the design from the top level when required. In the example below, numbered as a sub-number of the top level.

Example:

The following design is for a program that will read the name, prelim mark and coursework mark for a class of 20 pupils from a file. It will calculate a percentage from each of their prelim marks and coursework marks added together. It will then display the name of the pupil with the highest percentage and their percentage.

Main Steps	**Data Flow**
1 Get results	(**OUT**: pupil name(), prelim mark(), course mark())
2 Calculate percentages	**(IN**: prelim mark(), course mark() **OUT**: percentage())
3 Find position of pupil with top mark	(**IN**: percentage() **OUT**: top position)
4 Display pupil with top mark	(**IN**: pupil name(), top position)

Refinements

1.1 Open marks file
1.2 Start fixed loop for each pupil
1.3 Get pupil name()
1.4 Get prelim mark()
1.5 Get course mark()
1.6 End fixed loop
1.7 Close marks file

2.1 Start fixed loop for each pupil
2.2 percentage() equals (prelim mark() + course mark()) divided by 1.5
2.3 End fixed loop

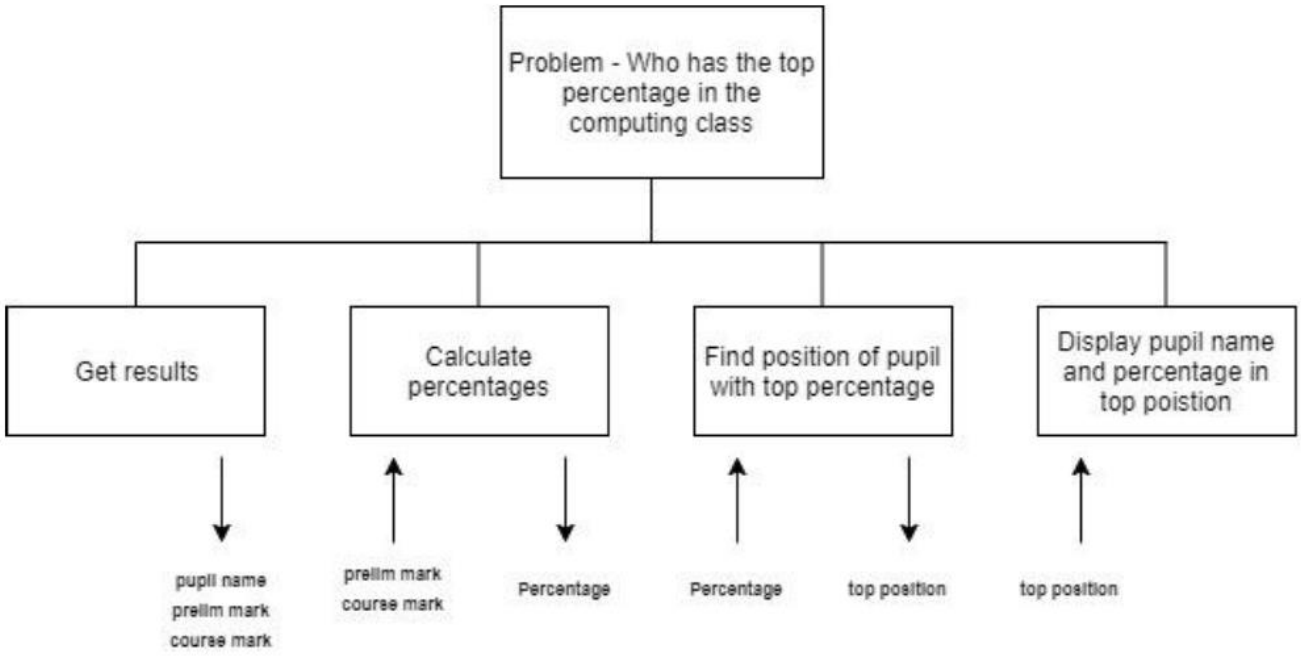
3.1 top position equals first position
3.2 Start fixed loop from second pupil
3.3 If percentage() is greater than current top percentage Then
3.4 set position as new top position
3.5 End If
3.6 End fixed loop

4.1 Display “Top pupil is”, pupil name(top position), “with”, percentage(top position), “percent”

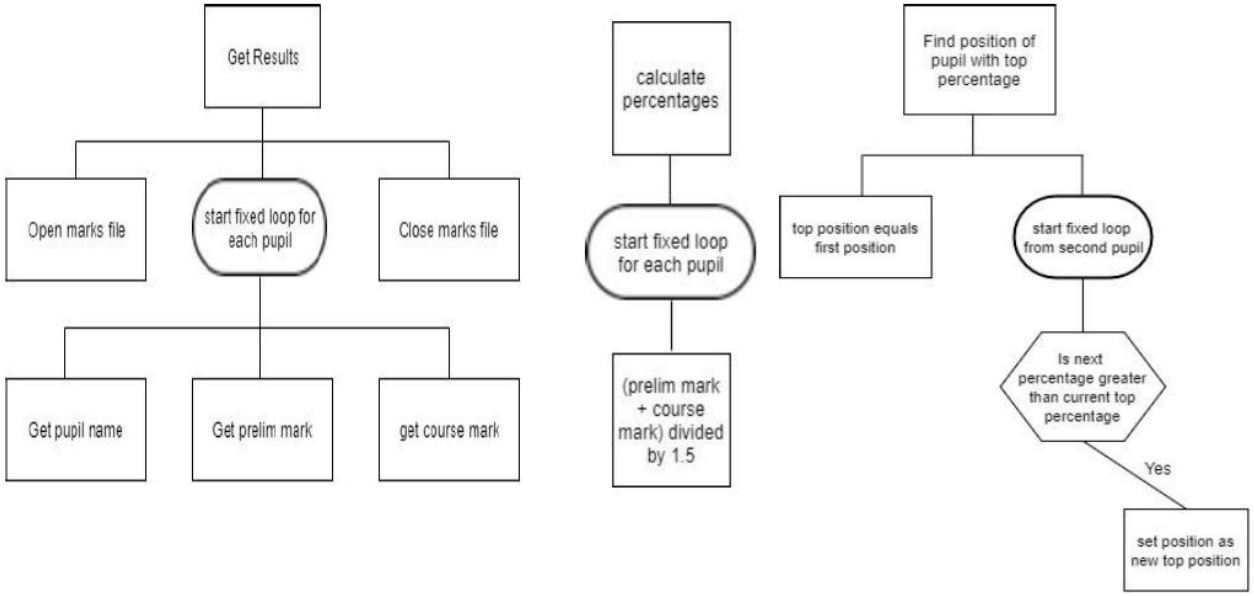
Structure Diagrams

The following structure diagram solves the same problem as the pseudocode:

- **Top level design** — the major steps of the design.
- **Data flow** — shows the information that must flow In or Out from the sub-programs. In the example below, written underneath the top level design with an arrow showing whether they are in or out.



- **Refinements** — break down the design from the top level into smaller steps. They can be shown separately from the top level design or below the top level design.



WIREFRAME (USER INTERFACE DESIGN)

The design of the user interface (the visual layout that allows the user to interact with the programming code) can be represented using a **wireframe** diagram.

A wireframe diagram is a visual representation of how the user interface will look and it will show the position of different elements such as text, graphics, navigation etc. It is also used as a visual representation to demonstrate the input and output of a program.

A wireframe diagram can be a detailed sketch or detailed image as shown below.

The wireframe diagram should clearly show

- **visual layout**
- **inputs**
- **validation**
- **underlying processes**
- **outputs**