

## Syllabus Content:

### 9.1. Computational Thinking Skills



Show an understanding of abstraction

#### Notes and guidance



Need for and benefits of using abstraction



Describe the purpose of abstraction,



Produce an abstract model of a system by only including essential details



Describe and use decomposition

#### Notes and guidance



Break down problems into sub-problems leading to the concept of a program module (procedure /function)

## Computational thinking and problem-solving:

Computational thinking is a problem-solving process where a number of steps are taken in order to reach a solution, rather than relying on rote learning to draw conclusions without considering these conclusions.

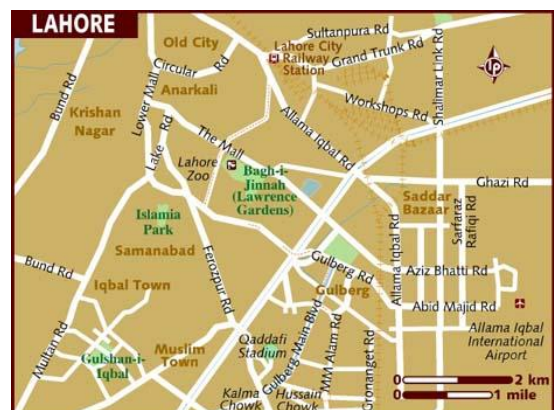
Computational thinking involves abstraction, decomposition, data-modelling, pattern recognition and algorithm design.

### Abstraction:

Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. Abstraction involves filtering out information that is not necessary to solving the problem.

Abstraction encourages the development of simplified models that are suited to a specific purpose by eliminating any unnecessary characteristics from that model. Many everyday items use Abstraction, such as maps, calendars and timetables etc.

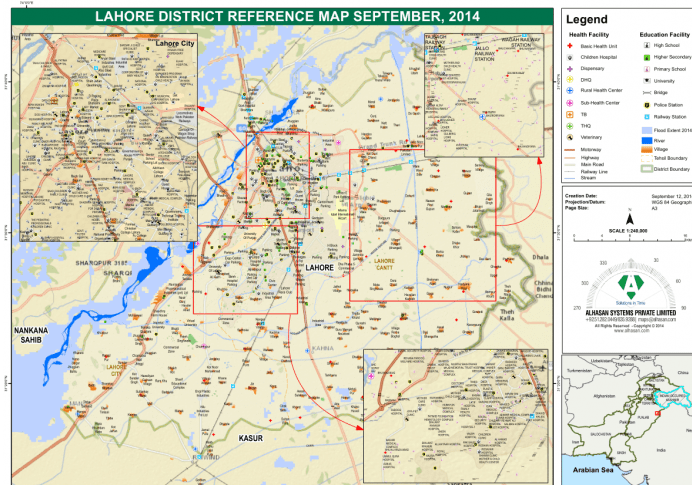
Lahore Map here is a road map of city. It uses Abstraction as it only shows roads to give drivers necessary details to drive from one place to other



Contact: 03004003666

Email: majidtahir61@gmail.com

Whereas other map of city Lahore show all details:






Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, What happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.

**Abstraction** is a powerful methodology to manage complex systems. Abstraction is managed by well-defined objects and their hierarchical classification.

**For example** a car itself is a well-defined object, which is composed of several other smaller objects like a gearing system, steering mechanism, engine, which are again have their own subsystems. But for humans car is a one single object, which can be managed by the help of its subsystems, even if their inner details are unknown.

### Benefits of Abstraction:

Benefits of eliminating unnecessary details from model include:

-  Time required to develop a program reduces.
-  Program is smaller in size and takes less space.
-  Program works efficiently, and modification and maintenance is easier.

### Decomposition:

Decomposition is essential part of Computational Thinking. When rigorous decomposition is done, many simple programs are made which look complex in beginning.

Decomposition means breaking tasks down into smaller parts in order to explain a process more clearly.

Decomposition is another word for step-wise refinement.

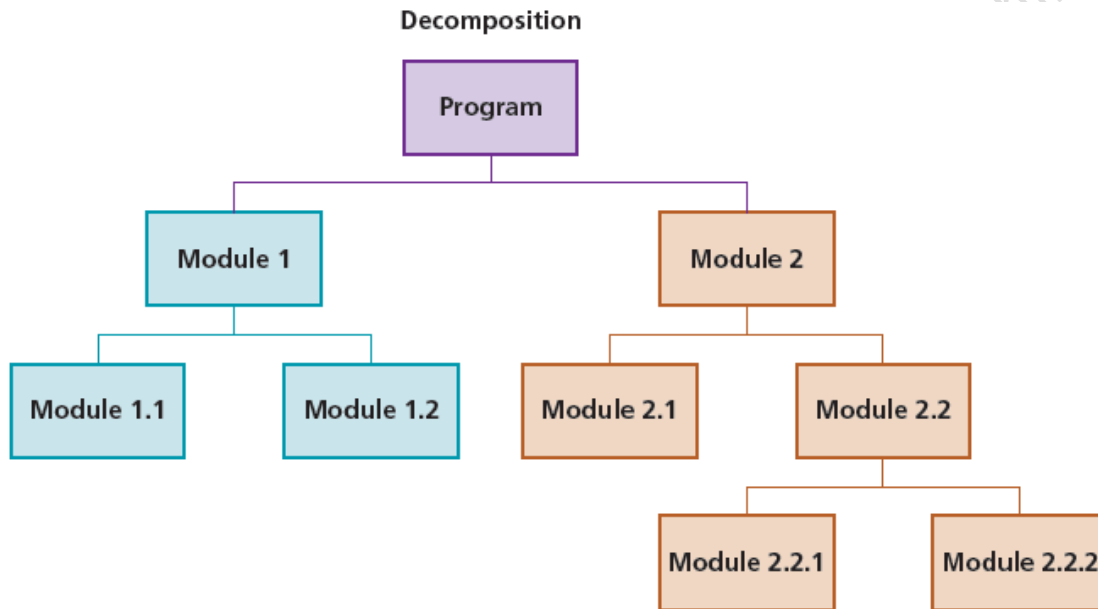
In structured programming, algorithmic **decomposition** breaks a process down into well-defined steps.

## Stepwise Refinement:

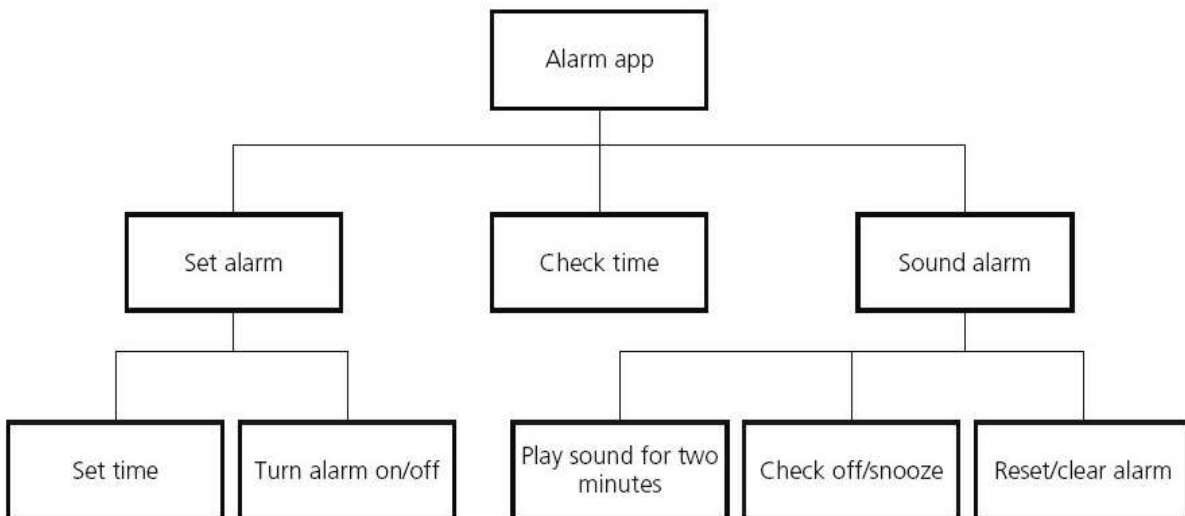
Algorithm done earlier were simple and short.

- When complex problems are solved, decomposition is used to break down problem into sub-problems and more manageable parts.
- These parts then need to be written as series of steps where each step can be written as a statement in high-level programming language.

The process of breaking down a problem into smaller sub-systems is called **stepwise refinement**



e.g



## Pattern recognition

Pattern recognition means looking for patterns or common solutions to common problems and exploiting these to complete tasks in a more efficient and effective way. There are many standard algorithms to solve standard problems, such as insertion sort or binary search.

**Pattern recognition** is used to identify those parts which are similar so that same solution (common code) can be used to solve the problem.

When program is written, each final part is defined as separate module and can be written and tested separately. Program modules already written and tested can be identified and reused.

## Algorithm design

Algorithm design involves developing step-by-step instructions to solve a problem

## Use subroutines to modularize the solution to a problem

### Subroutine/sub-program

A subroutine is a self-contained section of program code which performs a specific task and is referenced by a name.

A subroutine resembles a standard program in that it will contain its own local variables, data types, labels and constant declarations.

There are two types of subroutine. These are procedures and functions.

## Functions and Procedures

### Procedure

A procedure is a subroutine that performs a specific task without returning a value to the part of the program from which it was called.

### Function

A function is a subroutine that performs a specific task and returns a value to the part of the program from which it was called.

Note that a function is 'called' by writing it on the right hand side of an assignment statement.

### Parameter

A parameter is a value that is 'received' in a subroutine (procedure or function). The subroutine uses the value of the parameter within its execution.

The action of the subroutine will be different depending upon the parameters that it is passed. Parameters are placed in parenthesis after the subroutine name.

For example: Square(5) 'passes the parameter 5 – returns 25

## By Ref vs. By Val

Parameters can be passed by reference (byref) or by value (byval).

If you want to pass the value of the variable, use the ByVal syntax. By passing the value of the variable instead of a reference to the variable, any changes to the variable made by code in the subroutine or function will not be passed back to the main code. This is the default passing mechanism when you don't decorate the parameters by using ByVal or ByRef.

If you want to change the value of the variable in the subroutine or function and pass the revised value back to the main code, use the ByRef syntax. This passes the reference to the variable and allows its value to be changed and passed back to the main code.

## Example Program in VB – Procedures & Functions

```
Module Module1
    'this is a procedure
    Sub timestable(ByRef number As Integer)
        For x = 1 To 10
            Console.WriteLine(number & " x " & x & " = " & (number * x))
        Next
    End Sub

    'this is a function (functions return a value)
    Function adder(ByRef a As Integer, ByVal b As Integer)
        adder = a + b
        Return adder
    End Function

    Sub Main()
        timestable(7) 'this is a call (executes a procedure or function)
        timestable(3) 'this is a second call to the same procedure but now with different data
        timestable(9)
        Console.ReadKey()
        Console.Clear()

        Dim x As Integer
        x = adder(2, 3) 'call to function adder which returns a value
        Console.WriteLine("2 + 3 = " & x)
        Console.WriteLine("4 + 6 = " & adder(4, 6)) 'you can simply then code by
        putting the call directly into the print statement
        Console.ReadKey()
    End Sub
End Module
```

### References:

Computer Science AS & A Level Coursebook by Sylvia Langfield & Dave Duddell  
Computer Science AS & A Level by David Watson and Helen Williams