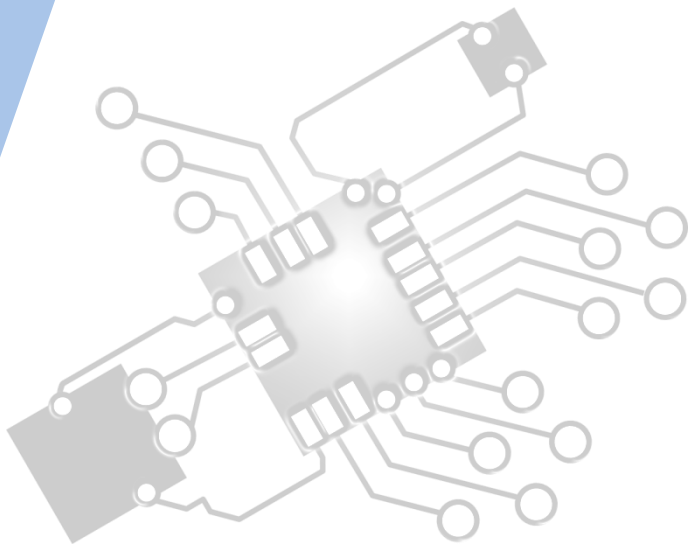
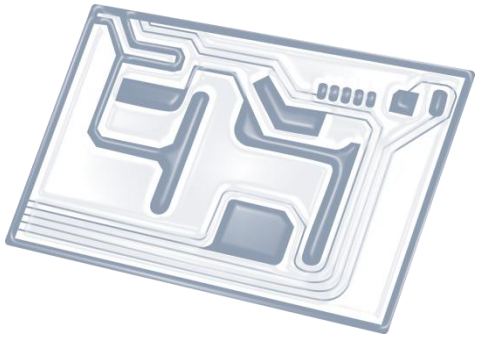
Abstract Data Structures

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL only 5 Overview

Thinking recursively
5.1.1 Identify a situation that requires the use of recursive thinking
5.1.2 Identify recursive thinking in a specified problem solution
5.1.3 Trace a recursive algorithm to express a solution to a problem
Abstract data structures
5.1.4 Describe the characteristics of a two-dimensional array
5.1.5 Construct algorithms using two-dimensional arrays
5.1.6 Describe the characteristics and applications of a stack
5.1.7 Construct algorithms using the access methods of a stack
5.1.8 Describe the characteristics and applications of a queue
5.1.9 Construct algorithms using the access methods of a queue
5.1.10 Explain the use of arrays as static stacks and queues
Linked lists
5.1.11 Describe the features and characteristics of a dynamic data structure
5.1.12 Describe how linked lists operate logically
5.1.13 Sketch linked lists (single, double and circular)
Trees
5.1.14 Describe how trees operate logically (both binary and non-binary)
5.1.15 Define the terms: parent, left-child, right-child, subtree, root and leaf
5.1.16 State the result of inorder, postorder and preorder tree traversal
5.1.17 Sketch binary trees
Applications
5.1.18 Define the term dynamic data structure
5.1.19 Compare the use of static and dynamic data structures
5.1.20 Suggest a suitable structure for a given situation

1: System design

2: Computer Organisation

3: Networks

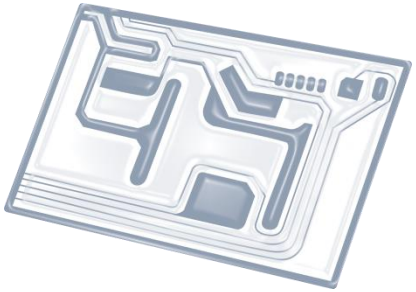
4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

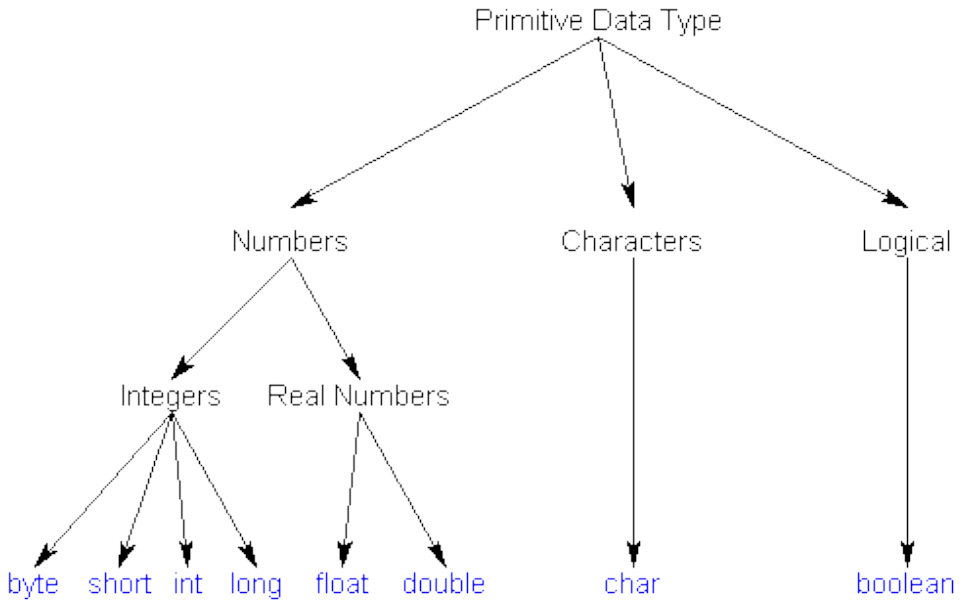


Topic 5.1.14

Describe how **trees** operate **logically** (both binary and non-binary)

Abstract Data Structures (ADTs)

- 2D array
- Stack
- Queue
- Linked List
- **(Binary) Tree**
- Recursion

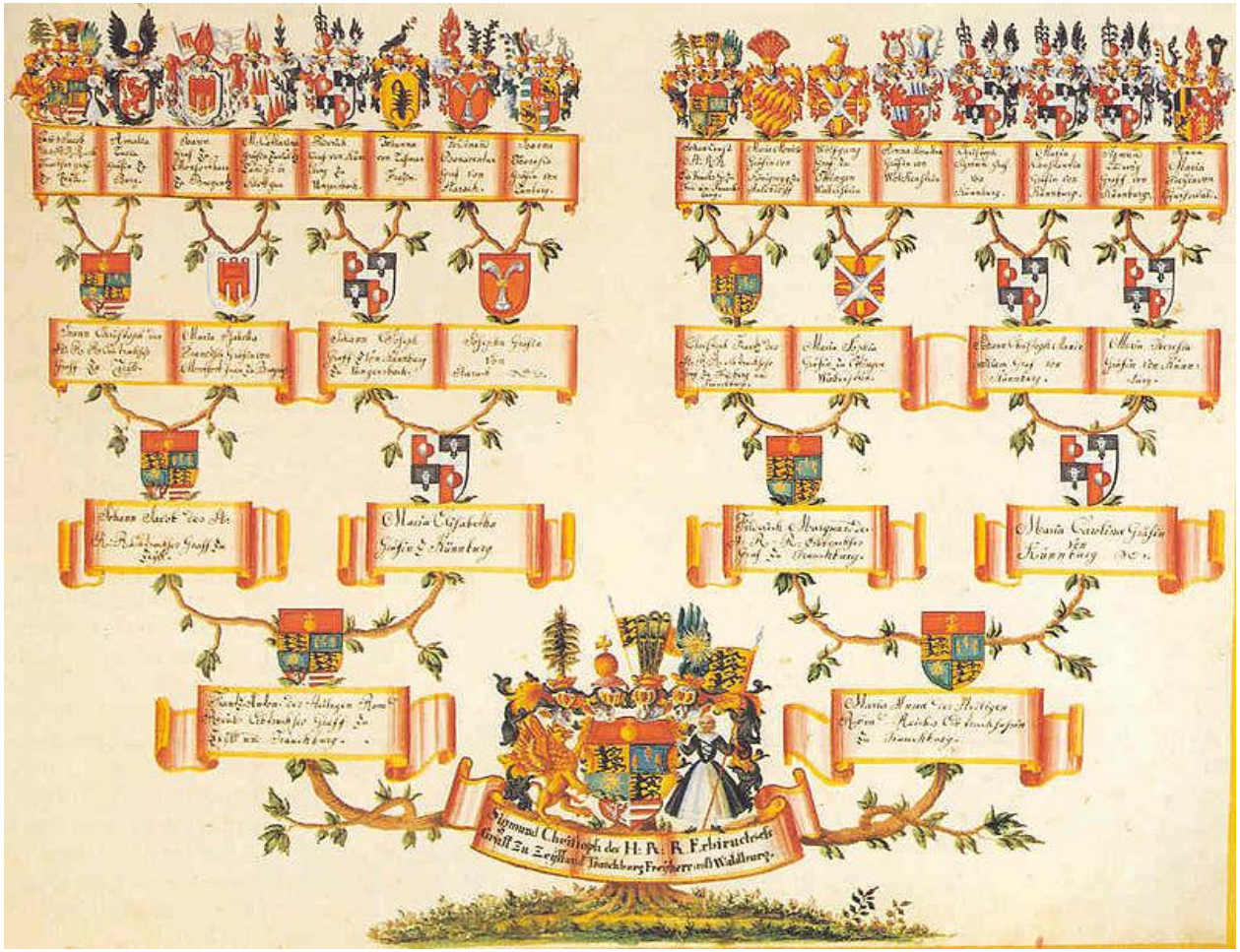


Important exam note:

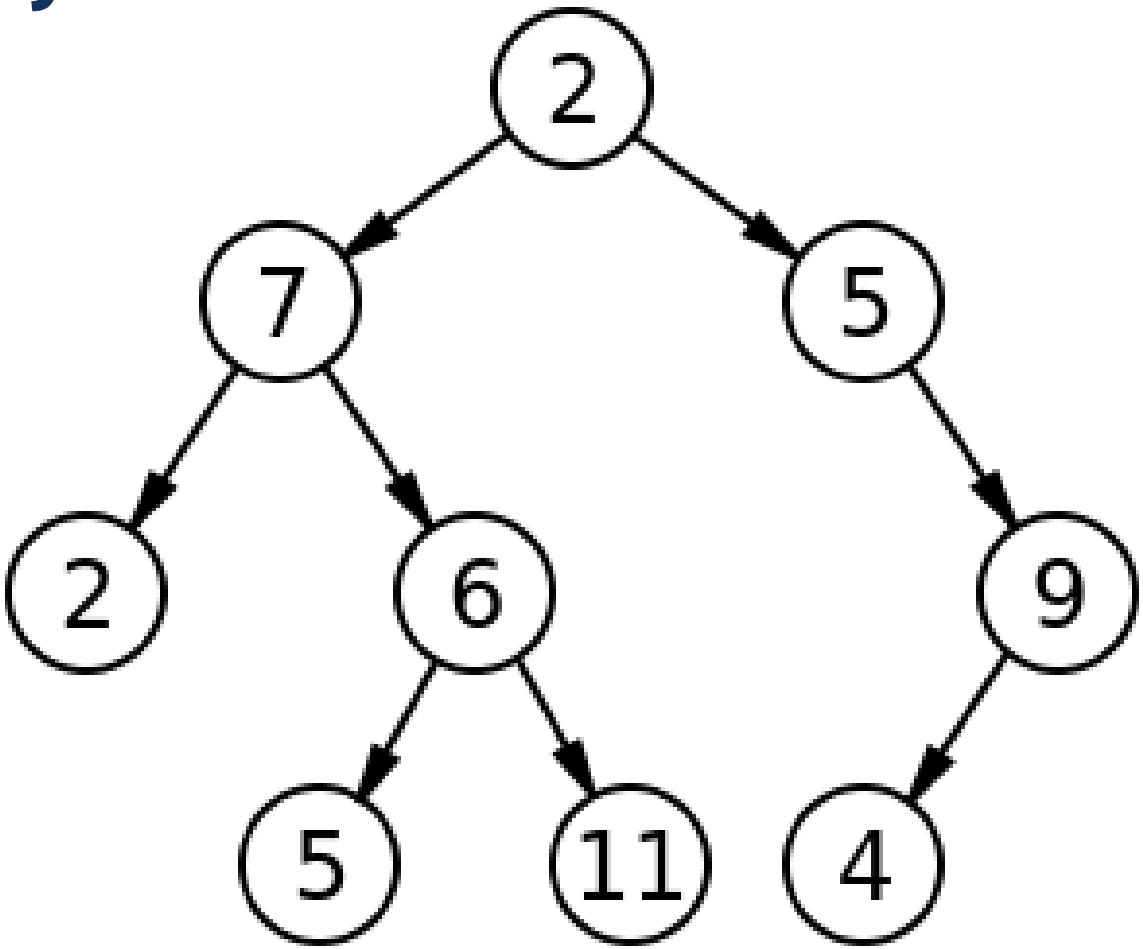
"Binary trees will be examined at the level of diagrams and descriptions. Students are not expected to construct tree algorithms using pseudocode."

"Tracing and constructing algorithms are not expected."

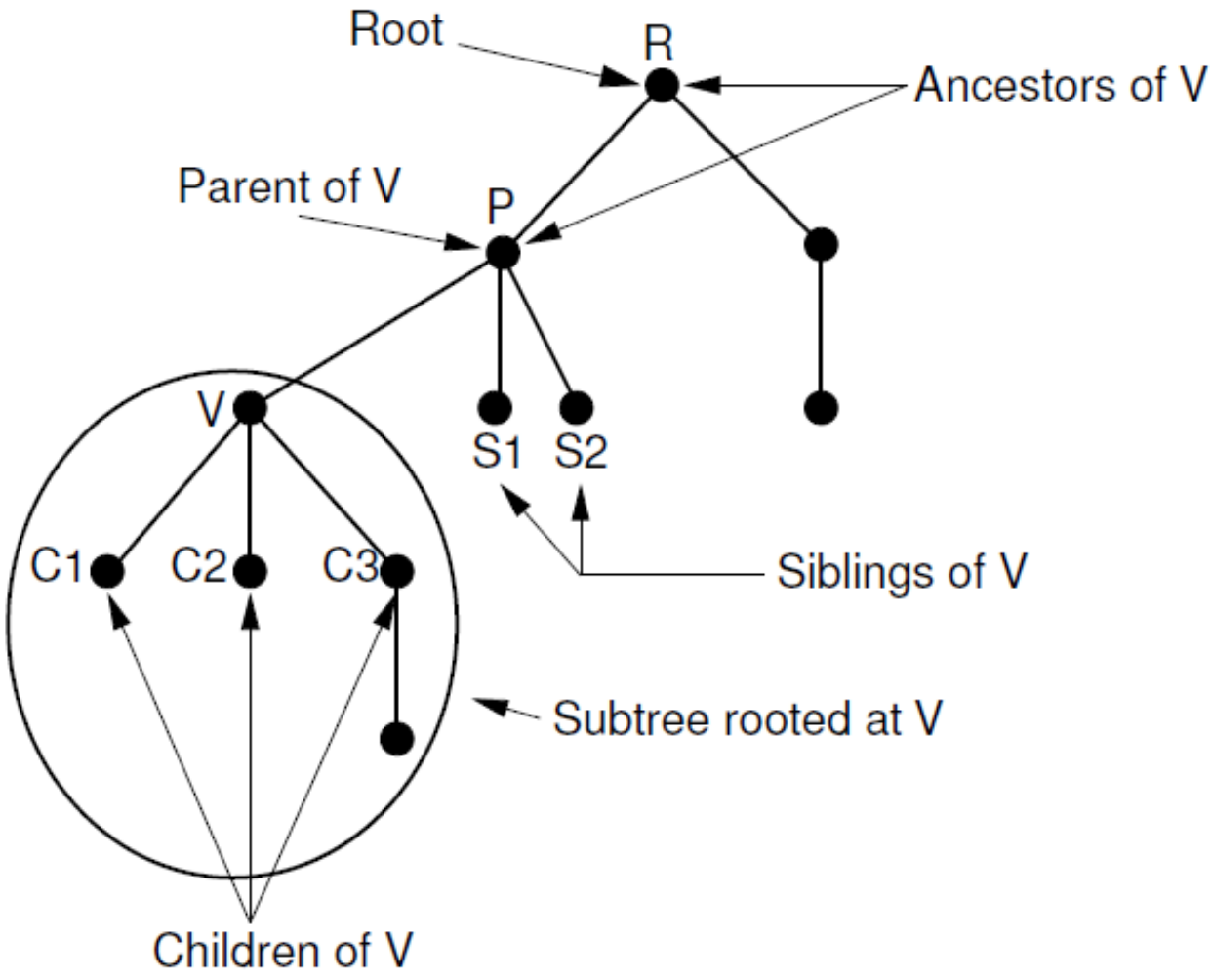
Binary Tree: Old Concept → New Name



Binary Tree



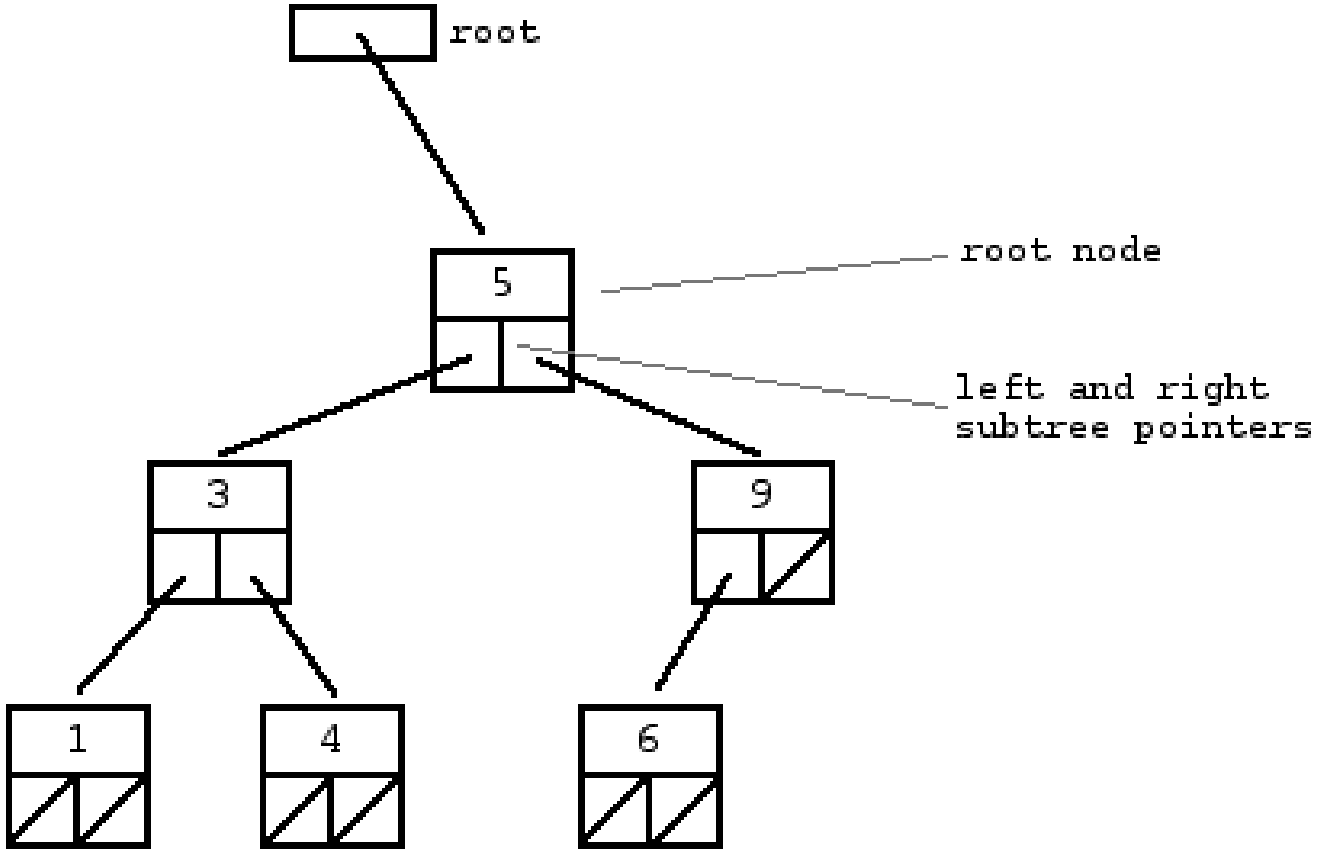
Non-binary tree (not in syllabus!)



More formally...

- A binary tree is made of **nodes**, where each node contains a "**left**" pointer, a "**right**" pointer, and a data element.
- The "**root**" pointer points to the topmost node in the tree.
- The left and right pointers recursively point to smaller "**subtrees**" on either side.
- A **null pointer** represents a binary tree with **no elements** -- the empty tree.
- **The formal recursive definition is**: a **binary tree** is either empty (represented by a null pointer), or is made of a single node, where the left and right pointers (recursive definition ahead) each point to a **binary tree**.

Formal visualisation



See also: http://cslibrary.stanford.edu/110/BinaryTrees.html