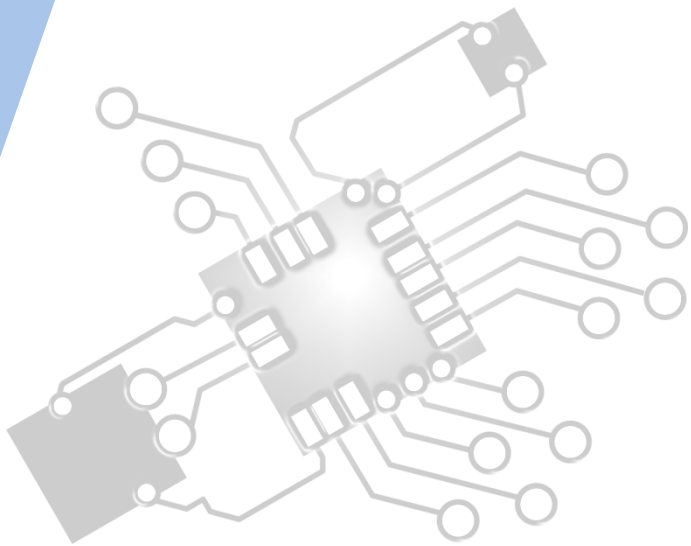
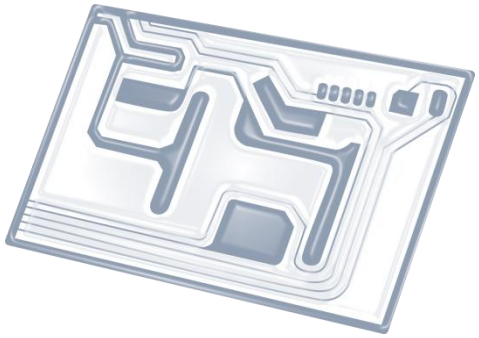
Planning & system installation

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

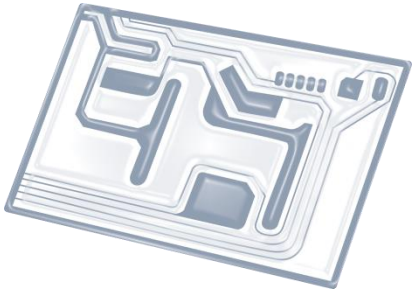
4: Computational thinking

5: Abstract data structures

6: Resource management

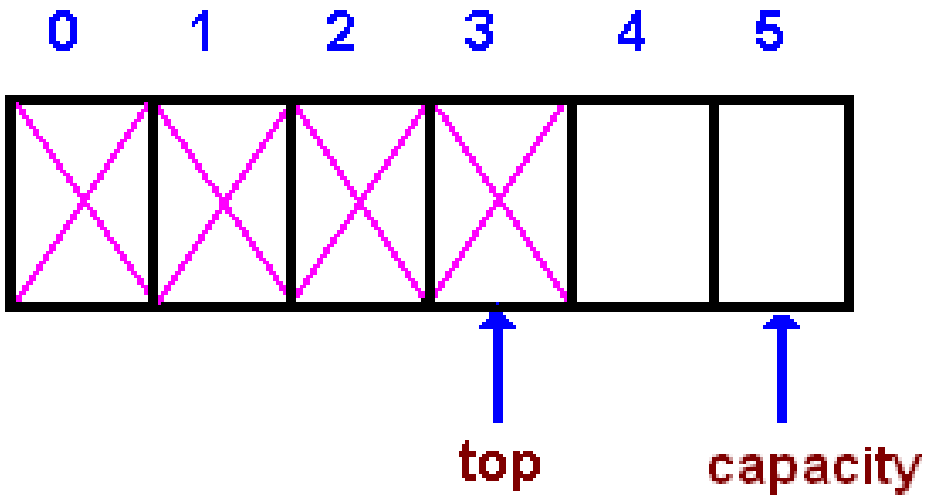
7: Control

D: OOP



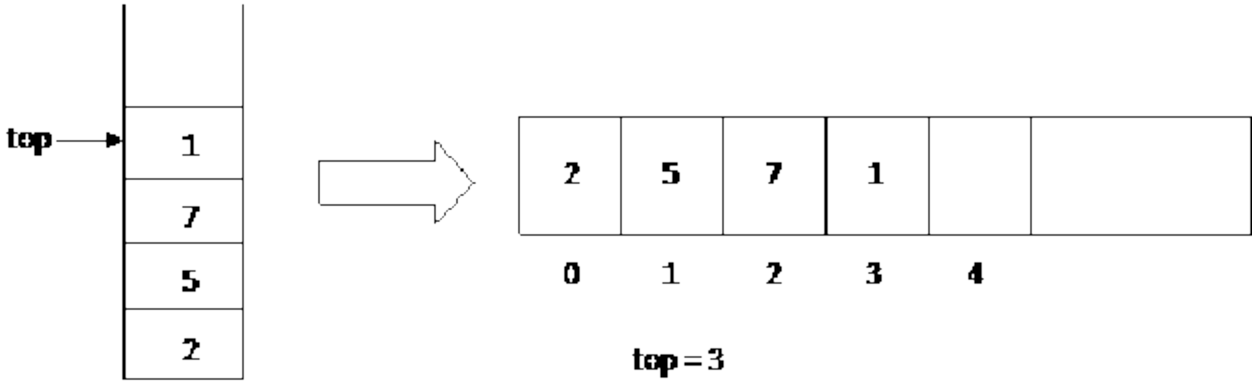
Topic 5.1.10

Explain the use of **arrays** as **static stacks** and **arrays**



Using Arrays for Stacks/Queues

- Arrays are **static** data structures (can't change size)
- Stacks & queues are **dynamic** data structures (can change size)
- You can use an array to behave like a stack or a queue, provided you give **it enough elements** to allow the stack or queue to function.



Useful explanation

See this explanation about how arrays can be used for stacks (and all the problems that brings with it!)

https://www.cs.bu.edu/teaching/c/stack/array/

Stack - Array Implementation

1. **Abstract idea of a stack:**

The *stack* is a very common data structure used in programs. By *data structure*, we mean something that is meant to *hold* data and provides certain *operations* on that data.

One way to describe how a stack data structure behaves is to look at a physical analogy, a stack of books...

Now, a *minimal* set of things that we might do with the stack  are the following:

Place a book on the top...  **Take one off the top...**  **See if the stack is empty...**  **NOT Empty**, there are books on the stack!

We'll consider other, more complex operations to be inappropriate for our stack. For example, pulling out the 3rd book from the top cannot be done *directly* because the stack might fall over.

How might you get the 3rd book from the top using only the simple operations?

From the IBO:

"Students should be able to explain push and pop operations, and test for an empty/full stack"

AND

"Students should be able to explain enqueue and dequeue operations, and test on empty/full queue"