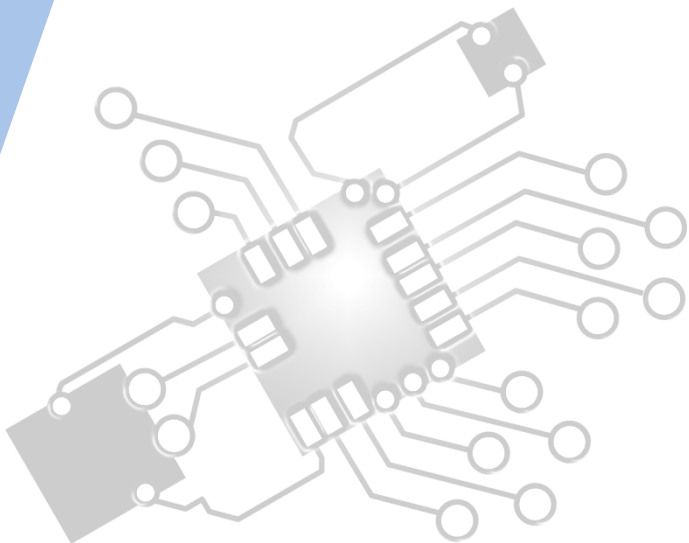




# **Computational thinking, problem-solving and programming:**

*Connecting computational thinking and program design*

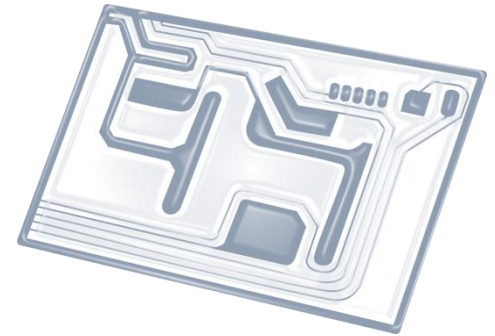
## **IB Computer Science**



*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 & SL 4.2 Overview

- 4.2.1 Describe the characteristics of standard algorithms on linear arrays
- 4.2.2 Outline the standard operations of collections
- 4.2.3 Discuss an algorithm to solve a specific problem
- 4.2.4 Analyse an algorithm presented as a flow chart
- 4.2.5 Analyse an algorithm presented as pseudocode
- 4.2.6 Construct pseudocode to represent an algorithm
- 4.2.7 Suggest suitable algorithms to solve a specific problem
- 4.2.8 Deduce the efficiency of an algorithm in the context of its use
- 4.2.9 Determine the number of times a step in an algorithm will be performed for given input data



1: System design

2: Computer Organisation



3: Networks

4: Computational thinking



5: Abstract data structures

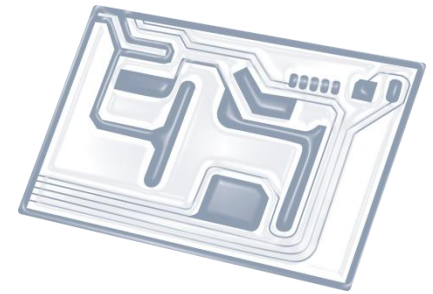
6: Resource management



7: Control

D: OOP

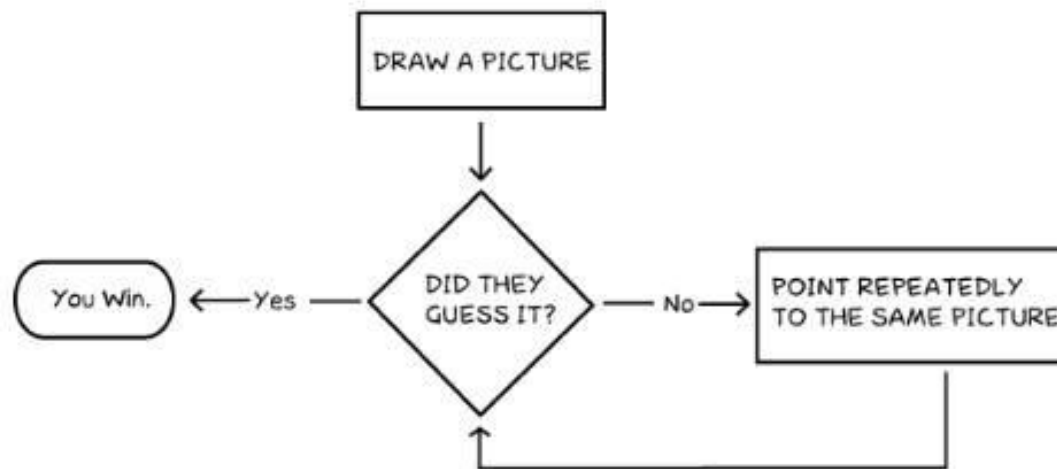




# Topic 4.2.4

Analyse an **algorithm** presented as a flow chart

## How To Play Pictionary



# Teacher's notes:

- Examination questions may involve **variables, calculations, simple and nested loops, simple conditionals and multiple or nested conditionals.**
- This would include **tracing an algorithm** as well as assessing its **correctness.**
- Students will not be expected to construct a flow chart to represent an algorithm in an exam.



# Make sure you know the symbols

These flowchart shapes are internationally recognised, so we must use them and **NOT** invent our own ones!



**Terminator;** This either contains START or END, and only one of each exists in a flowchart. They specify where the start and end of a flowchart is.



**Input/output;** We use this shape to show that something is going IN or OUT of the system we are designing. For example, we put a tea bag into a cup.

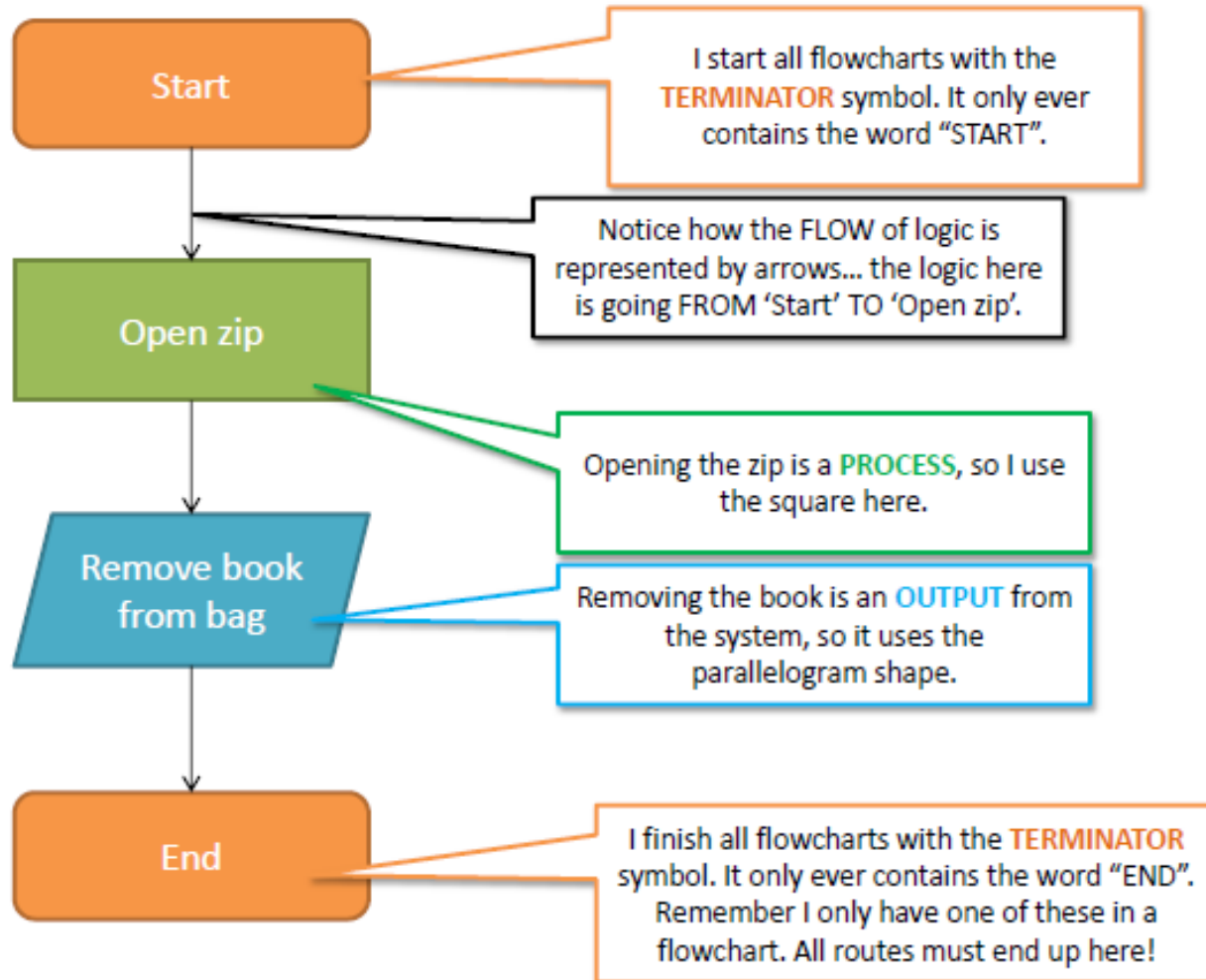


**Process;** We use this to show that something is happening. So, if I was to walk five steps forward, that is a process. It can't be an output or input as nothing is going in or out!

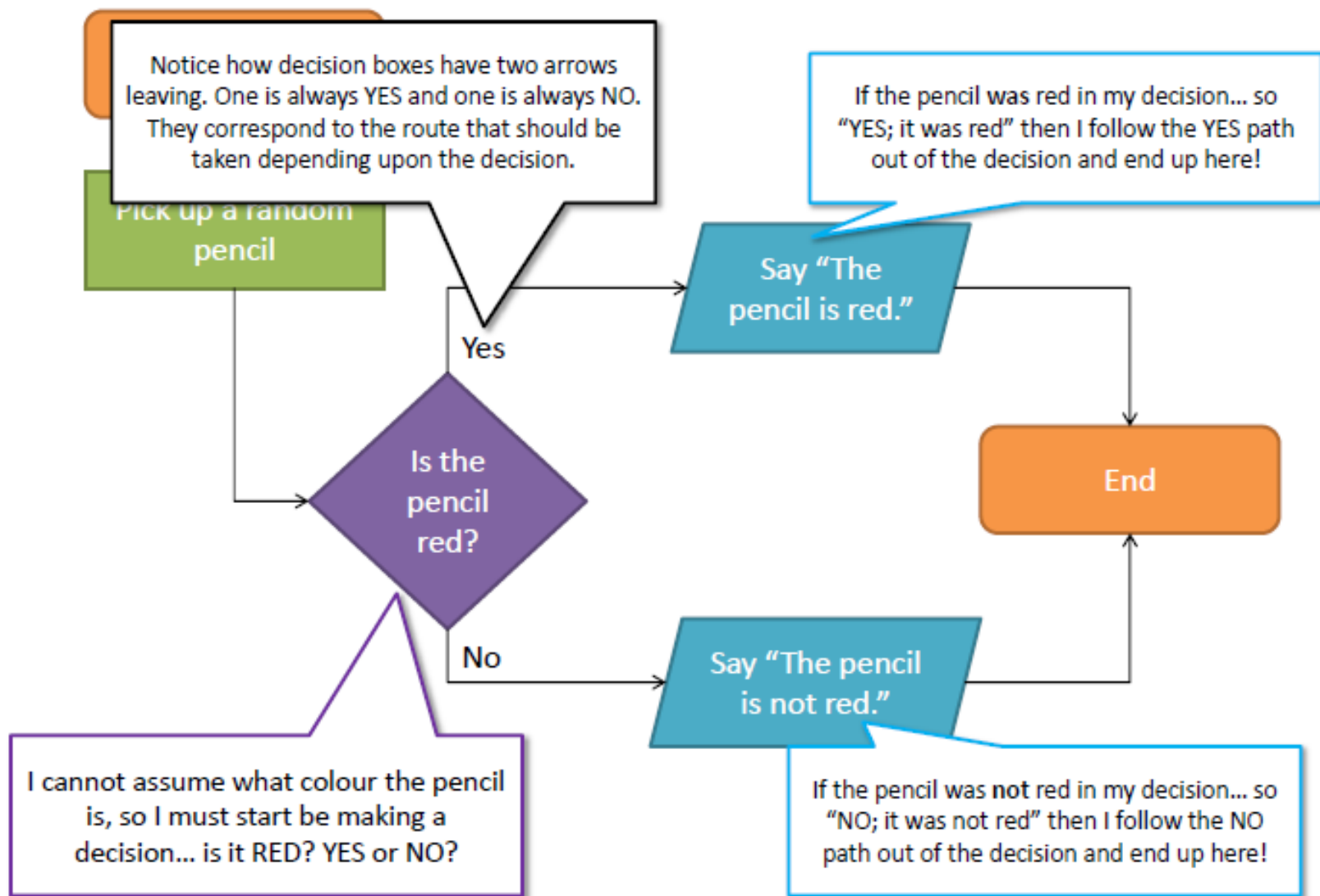


**Decision;** Decisions are used when we need to make a choice. Decisions **MUST** have two exits, one labelled YES and one NO. They are the only shape that has two exits. For example, "Is the kettle boiled?" This is either a YES or a NO...

## Example 1 - Getting a book out of a backpack. Explained

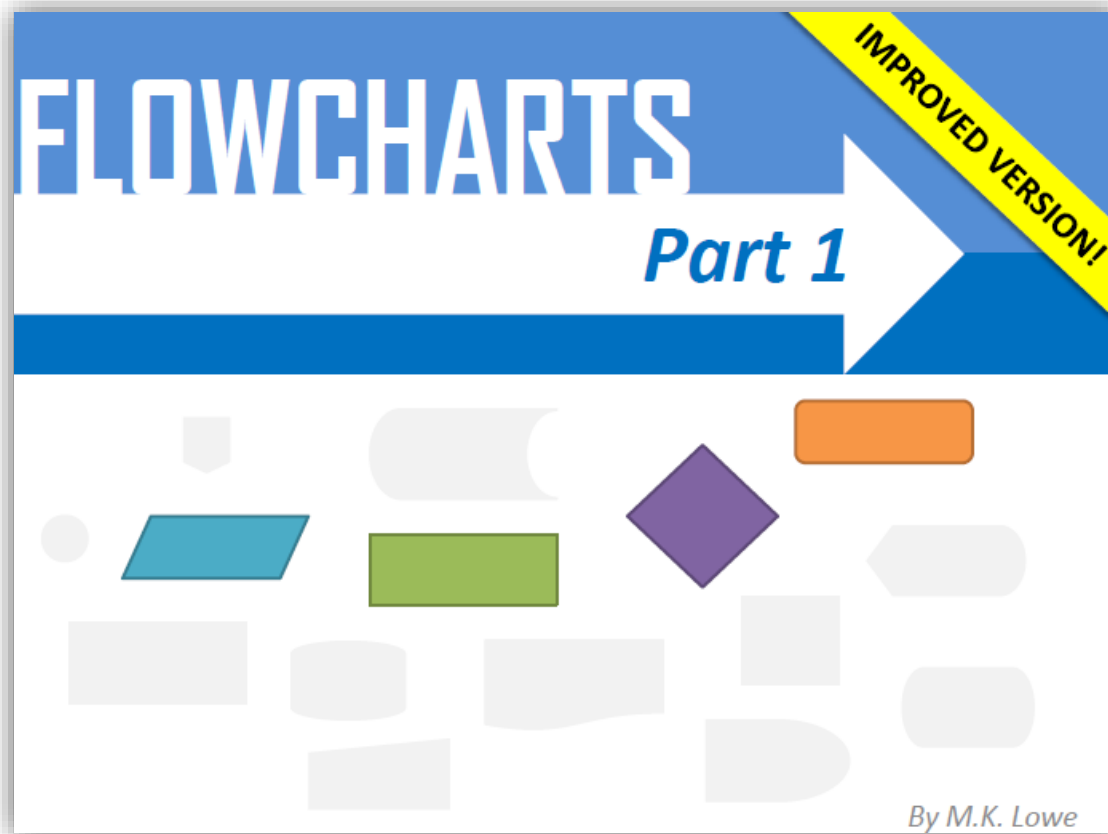


## Example 2 – Determine if I have picked up a red pencil. Explained



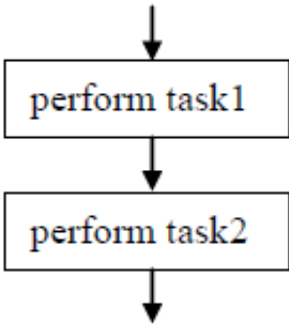
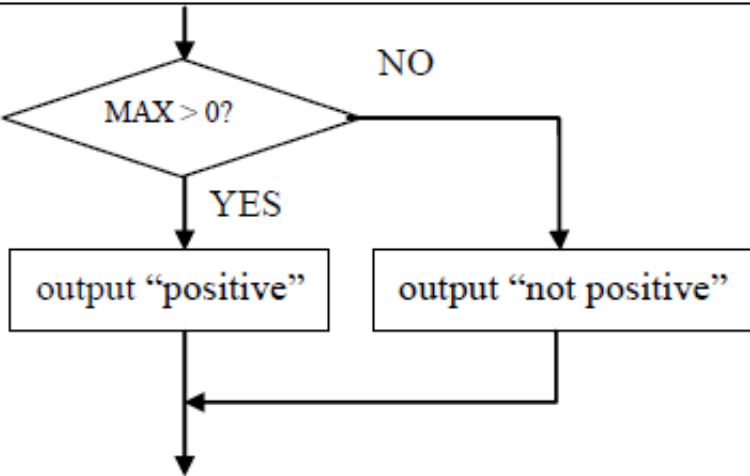


# See the **flowchart notes** on IB CompSci

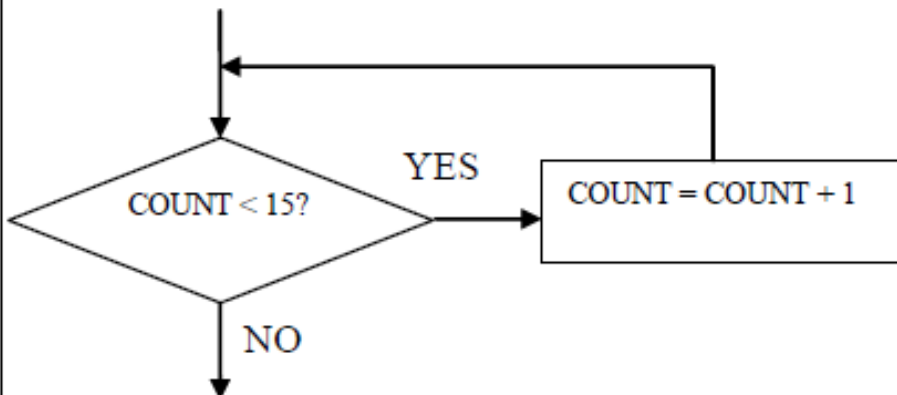


# Officially from the IB...

You get this in the exam

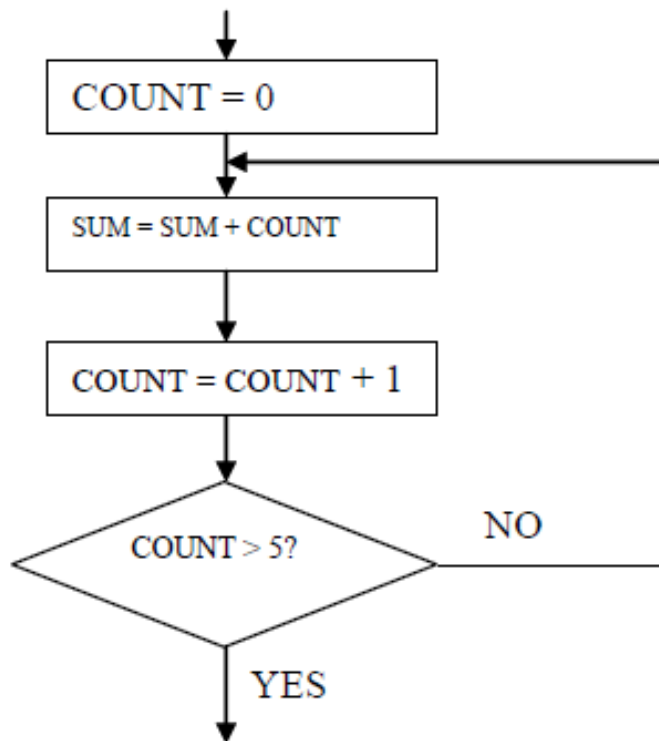
Operation	Flowchart example	Pseudocode example
<b>sequential operations</b>	 <pre> graph TD     Start(( )) --&gt; Task1[perform task1]     Task1 --&gt; Task2[perform task2]     Task2 --&gt; End(( ))         </pre>	<pre> perform task1 perform task2         </pre>
<b>conditional operations</b>	 <pre> graph TD     Start(( )) --&gt; Decision{MAX &gt; 0?}     Decision -- YES --&gt; OutputPos[output "positive"]     Decision -- NO --&gt; OutputNotPos[output "not positive"]     OutputPos --&gt; End(( ))     OutputNotPos --&gt; End         </pre>	<pre> if MAX &gt; 0 then     output "positive" else     output "not positive" end if         </pre>

**while-loop**

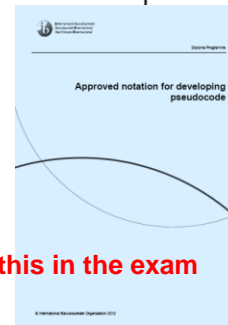


loop while COUNT < 15  
 COUNT = COUNT + 1  
 end loop

**from/to-loop**



loop COUNT from 0 to 5  
 SUM = SUM + COUNT  
 end loop



**You get this in the exam**