GCSE COMPUTING
KNOWLEDGE ORGANISER

2.1 COMPUTATIONAL THINKING ALGORITHMS AND PROGRAMMING

2.1.1 COMPUTATIONAL THINKING

Principles of computational thinking:

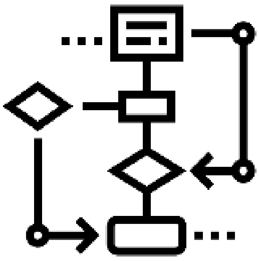
- ❑ Abstraction
- ❑ Decomposition
- ❑ Algorithmic thinking

REVISION NOTE
Pattern recognition is one of the four methods of computational thinking – but it is not studied at GCSE level

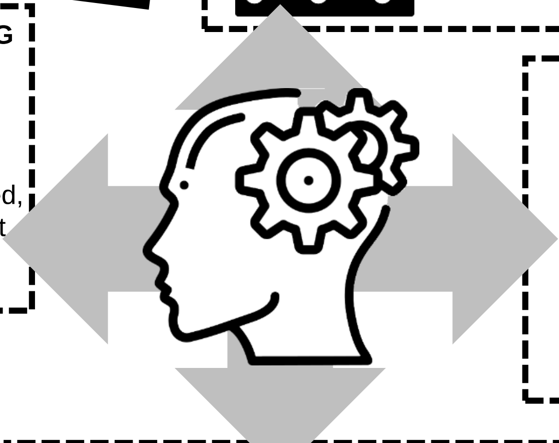
"The process of approaching problems systematically and creating solutions that can be carried out by a computer" ✓

"Thinking like a computer" ✗

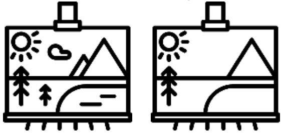
PATTERN RECOGNITION involves looking for similarities or patterns in different aspects of the problem.



ALGORITHMIC THINKING
An algorithm is a series of steps necessary to complete a task or solve a problem. Once an algorithm has been planned, code can be written so that the problem can be solved using a computer.



ABSTRACTION – taking only the important and relevant data about the problem and discarding the unnecessary data.

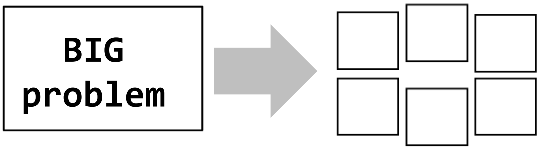


2.1.2 DESIGNING, CREATING AND REFINING ALGORITHMS

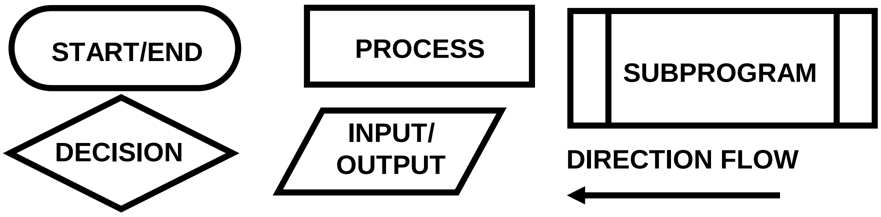
Identify the inputs, processes, and outputs for a problem
Structure diagrams
Create, interpret, correct, complete, and refine algorithms using:

- ❑ Pseudocode
- ❑ Flowcharts
- ❑ Reference language/high-level programming language
- ❑ Identify common errors
- ❑ Trace tables

DECOMPOSITION – taking a large problem and breaking it down into smaller, simpler problems. These are can be tackled more easily



FLOWCHARTS can be used to represent algorithms using the symbols shown here. Algorithms can also be represented using **PSEUDOCODE** or **REFERENCE LANGUAGE**



```
y = 0
z = 0
FOR x in range (4):
        y = x * 2
        z = z + y
```

TRACE TABLES are used to test and identify the outcome of algorithms

x	y	z
0	0	0
1	2	2
2	4	6
3	6	12
4	8	20

High Level Language	Pseudocode	Reference Language
Specific syntax must be used	No formal syntax – can take any form	More formal structure than pseudocode
Used to write code	Used to present an algorithm so that a human can understand it	Used to present an algorithm to closely resemble code

`FOR loop in range(10):`
`PRINT(loop)`
	`Loop 10 times`
	`Print loop position`
	`End loop`
		`FOR loop = 1 to 10`
		`PRINT(loop)`
		`NEXT loop`

2.1.3 SEARCHING AND SORTING ALGORITHMS

Standard searching algorithms:

□ Binary search
□ Linear search

A **BINARY SEARCH** requires data to be sorted in order before it can be searched. A **LINEAR SEARCH** does not –the algorithm will look at every item in list until it either locates the data or reaches the end of the list. The binary search is the more efficient of the two

```
INPUT item to be searched for
found = False
numbers = [4,2,6,1,5,3]
REPEAT
   Compare item with current item in list
   IF current item is the item searched for then
             found = True
UNTIL end of list OR found = True
IF found = True
   PRINT (“Item found”)
ELSE
   PRINT (“Item not found”)
```

LINEAR SEARCH

BINARY SEARCH

We are searching for 6 in a sorted list	1	2	3	4	5	6	7	
List is split in two at the mid point	1	2	3	4	5	6	7	6 > 4 so discard items less than 4
List is split in two at the mid point				4	5	6	7	6 > 5 so discard items less than 5
List is split in two at the mid point						6	7	Item has been found

Standard sorting algorithms:

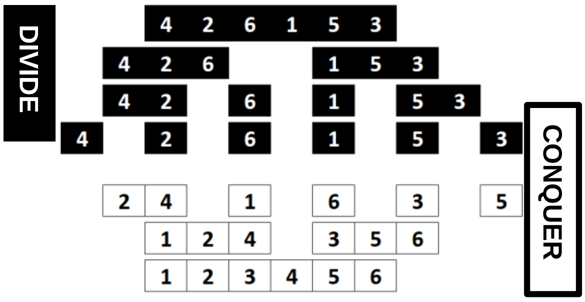
□ Bubble sort
□ Merge sort
□ Insertion sort

REVISION NOTE
You need to be familiar with searching sorting algorithms but there is no need for you to be able to code them

-A **BUBBLE SORT** is an algorithm for sorting data.
-The algorithm works by going through a list of unordered data and evaluating the data in pairs.
-If two data items are in the wrong order they are exchanged.
-The algorithm then moves to the next pair.
-When the algorithm reaches the end of the data, the process will be repeated until all data has been sorted correctly. This might take **SEVERAL PASSES** through the data.

STARTING DATA	4	2	6	1	5	3	
Items 1 & 2	2	4	6	1	5	3	2 > 4 so SWAP
Items 2 & 3	2	4	6	1	5	3	4 < 6 NO SWAP
Items 3 & 4	2	4	1	6	5	3	1 < 6 so SWAP
Items 4 & 5	2	4	1	5	6	3	5 < 6 so SWAP
Items 2 & 3	2	4	1	5	3	6	3 < 6 so SWAP

-A **MERGE SORT** is a **DIVIDE AND CONQUER** algorithm;
-First of all, the items of data in a list are divided in half until each item is in a **SUBLIST** of one item.(This is the **DIVIDE** stage)
-The algorithm will then merge each sublist, after comparing and sorting them as appropriate.
-When all of the data has been merged back into a single list it will be in the correct order. (This is the **CONQUER** stage)
- Merge sorts are more efficient than bubble or insertion sorts.



-An **INSERTION SORT** is more efficient than a bubble sort.
-The insertion sort works in a similar way to sorting a hand of cards.
-The algorithm works by comparing the current data item with the other items in the list
- If the data item is in the wrong place, it is shifted to left until it is in the correct place.
- This continues until all the items of data are in the correct place.

Unsorted list	4	2	6	1	5	3
1 inserted at the front of the list	1	4	2	6	5	3
2 inserted at the front of the list	1	2	4	6	5	3
3 inserted at the front of the list	1	2	3	4	6	5
4 would be inserted (4 is already in the correct place)	1	2	3	4	6	5
5 inserted at the front of the list	1	2	3	4	5	6
6 would be inserted (6 is already in the correct place)	1	2	3	4	5	6