GCSE Unit 2.1 | Algorithms

Name:

Specification & learning objectives

- Computational thinking.
- Standard searching algorithms.
- Standard sorting algorithms.
- How to produce algorithms.
- Interpreting, correcting and completing algorithms.

Resources

PG Online text book page ref: 66-88

CraignDave videos for SLR 2.1

Abstraction

Abstraction means:

Example of an abstraction:

Real aeroplane:

Paper aeroplane:

Necessary features of a paper aeroplane:

Unnecessary features of a paper aeroplane:

Abstraction

Cat:	Cat icon:	Necessary features of the icon:	Unnecessary features of the icon:
Dog:	Dog icon:	Necessary features of the icon:	Unnecessary features of the icon:
Rabbit:	Rabbit icon:	Necessary features of the icon:	Unnecessary features of the icon:

Abstraction

A computer program that outputs whether a capital city in Europe is north or south of another capital city in Europe only needs to know the latitude of the two cities. The other detail is unnecessary. This is an example of abstraction: including the necessary detail and not including the unnecessary detail.

City	Latitude (N)
Dublin	53.3498
London	51.5074
Oslo	59.9139
Paris	48.8566
Madrid	40.4168

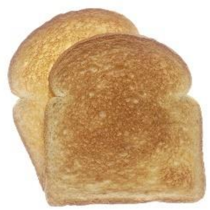
Program:

Decomposition

Decomposition means:

Examples of problem decomposition in every-day life:

Making toast:

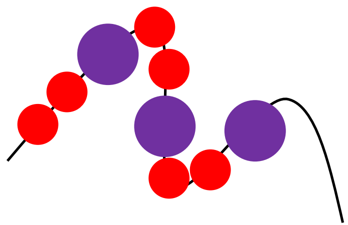


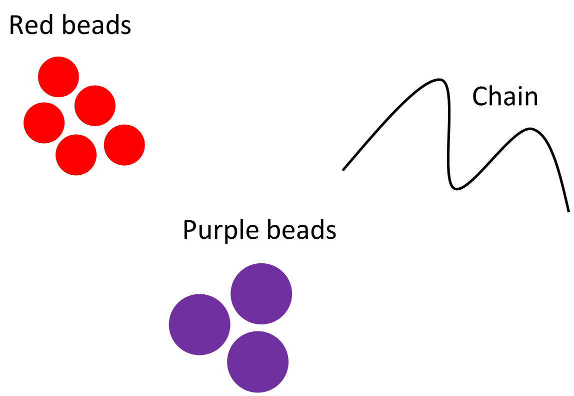
Making a fairy cake:

Decomposition

Advantages of decomposition include:

Example of problem decomposition in making costume jewellery:





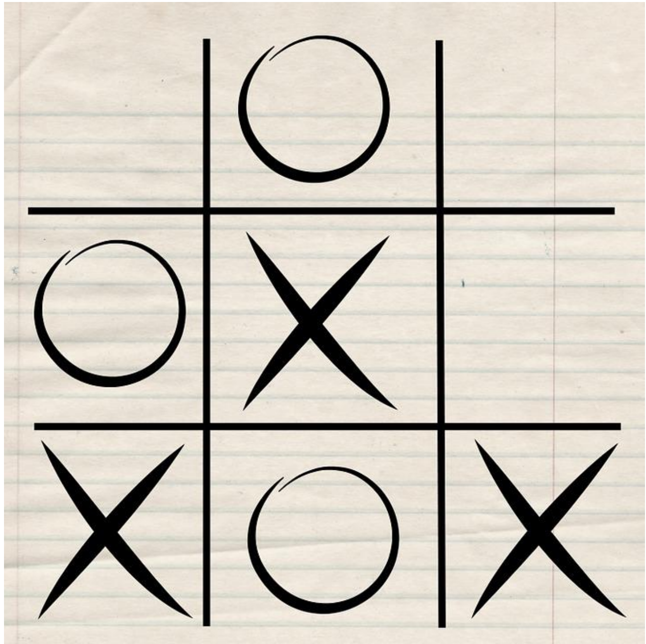
Algorithmic thinking

Decomposition of pick up sticks:

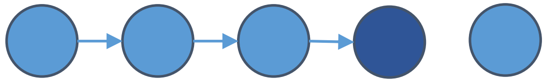
Program:

Algorithmic thinking

Decomposition of noughts and crosses:



Linear search



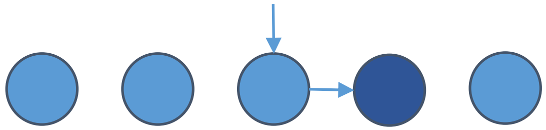
Explanation of a linear search:

Steps to find the Geography book on the shelf using a linear search:

Pseudocode of the linear search algorithm:

```
book = ["Archaeology", "Art", "Biology", "Chemistry", "Computing", "English", "French",
"Geography", "History", "Maths", "Psychology"]
```

Binary search



Explanation of a binary search:

Steps to find the Geography book on the shelf using a binary search:

Special condition for a binary search to work:

In most cases, the quicker search is performed by the: ______ algorithm. However, this is not true if the first item in the list is the one you want to find.

If the item you want to find is first in the list then the ______ algorithm would be quicker.

Binary search

Pseudocode of the binary search algorithm:

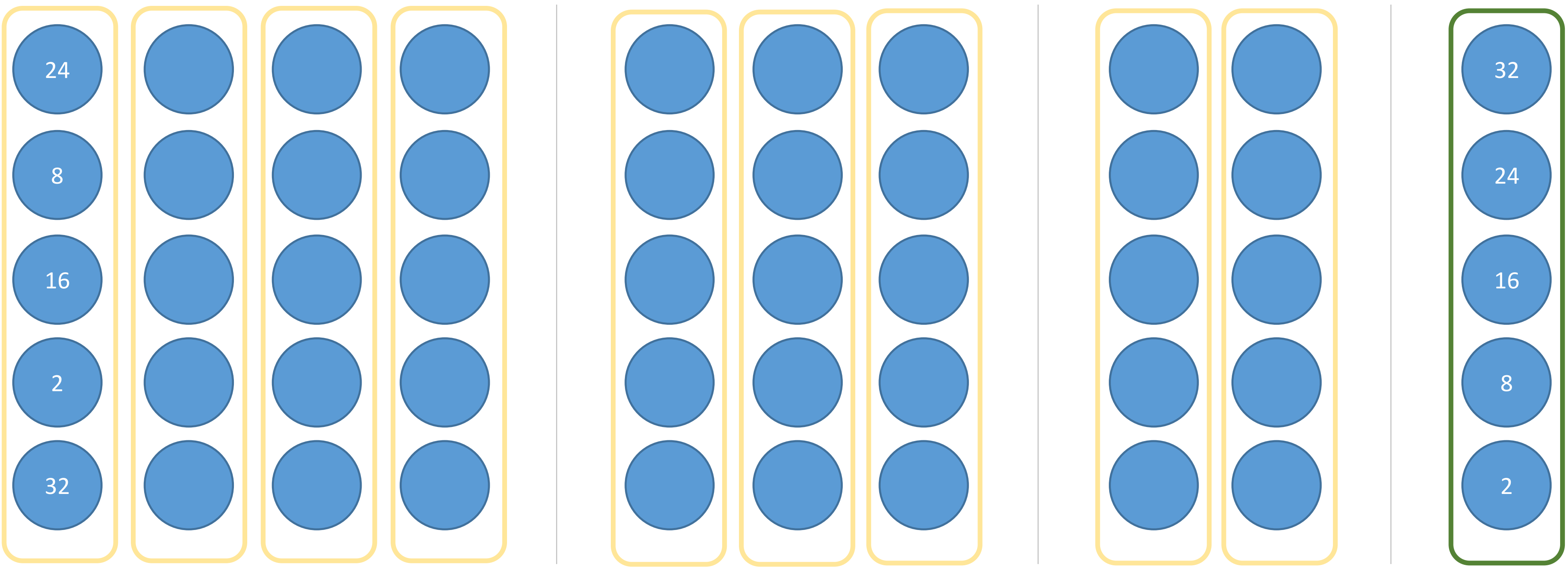
```
book = ["Archaeology", "Art", "Biology", "Chemistry", "Computing", "English", "French",
"Geography", "History", "Maths", "Psychology"]

found = False
left = 0
right = LEN(book)-1
find = "Geography"
```

Bubble sort

How a bubble sort works:

Note how 32 has "bubbled" to the top.
This is how the bubble sort got its name.

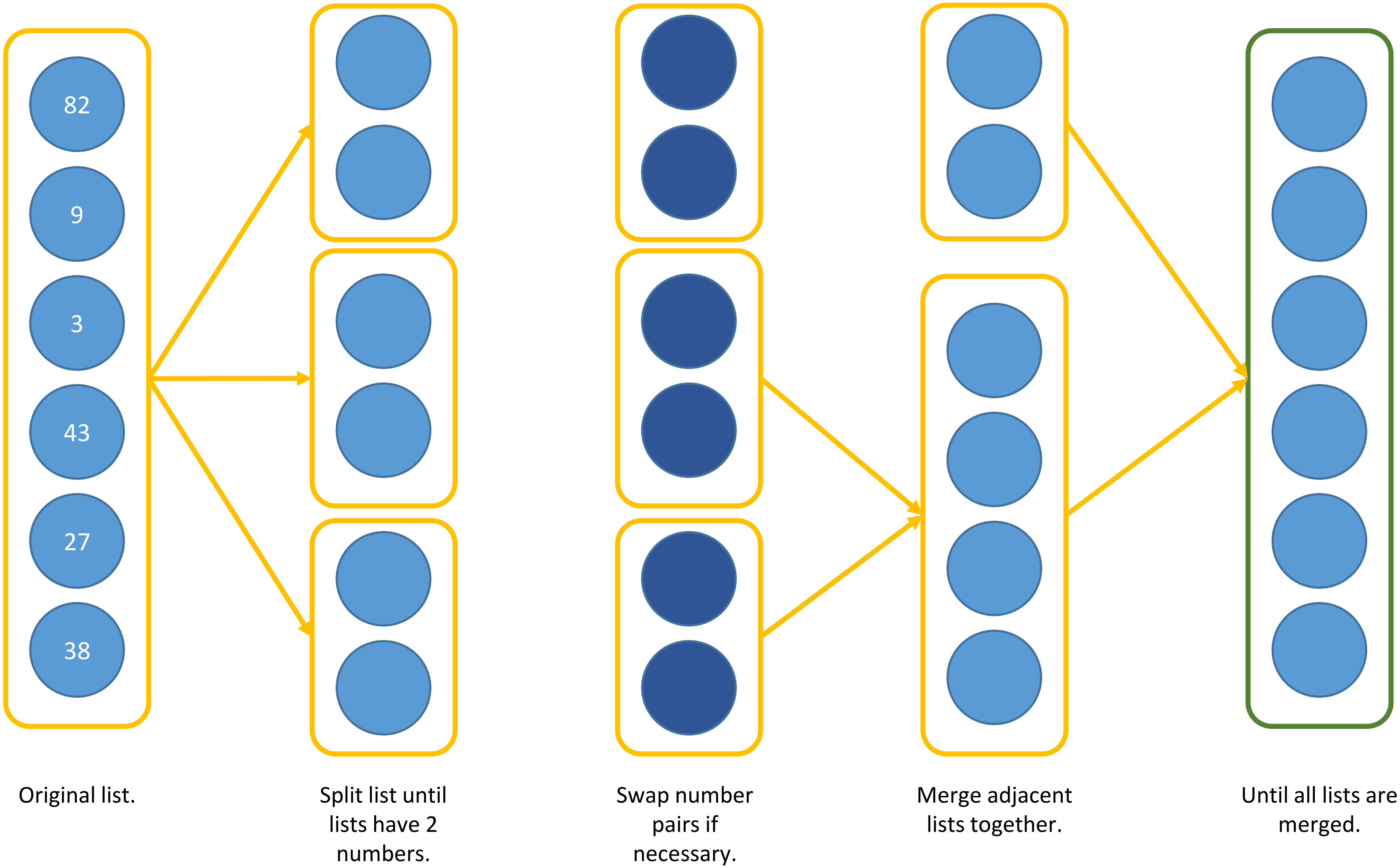
The algorithm has been optimised so it does not check the numbers already bubbled to the top. It can also stop if no swaps are made after all the numbers are checked.



Check 2 and 32. Swap	Check 32 and 16. Swap.	Check 32 and 8. Swap.	Check 32 and 24. Swap.	Check 2 and 16. No swap.	Check 16 and 8. Swap.	Check 16 and 24. No swap.	Check 2 and 8. No swap.	Check 8 and 16. No swap.	Check 2 and 8. No swap.

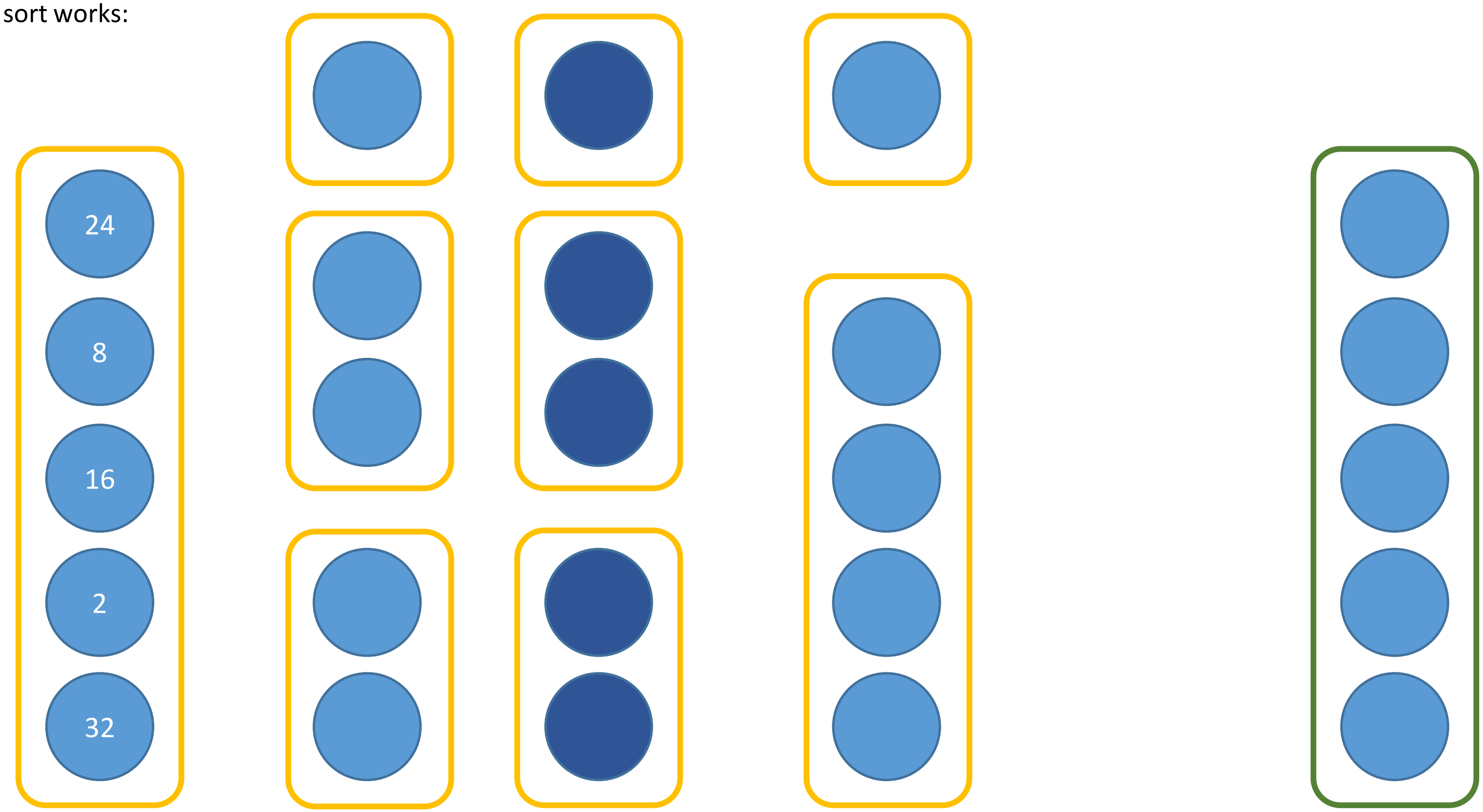
Merge sort

How a merge sort works:



Merge sort

How a merge sort works:



Original list.

Split into adjacent sub-lists of up to two numbers.

Swap numbers if necessary in each sub list. 8 and 16 swap.

Merge adjacent lists together by comparing the first number in each list, moving the smaller number into a new list, one number at a time.

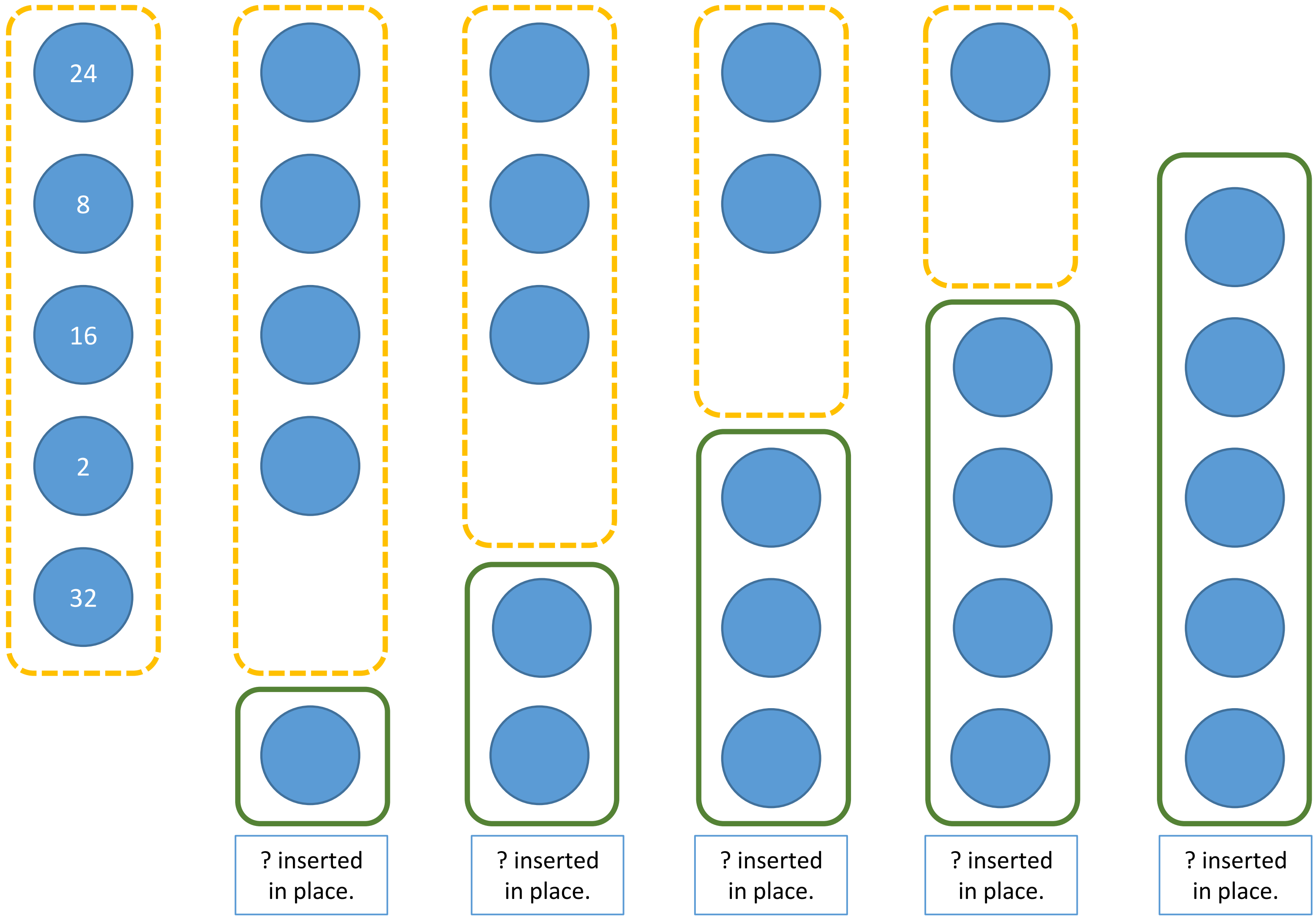
Merge adjacent lists together by comparing the first number in each list, moving the smaller number into a new list, one number at a time.

Insertion sort

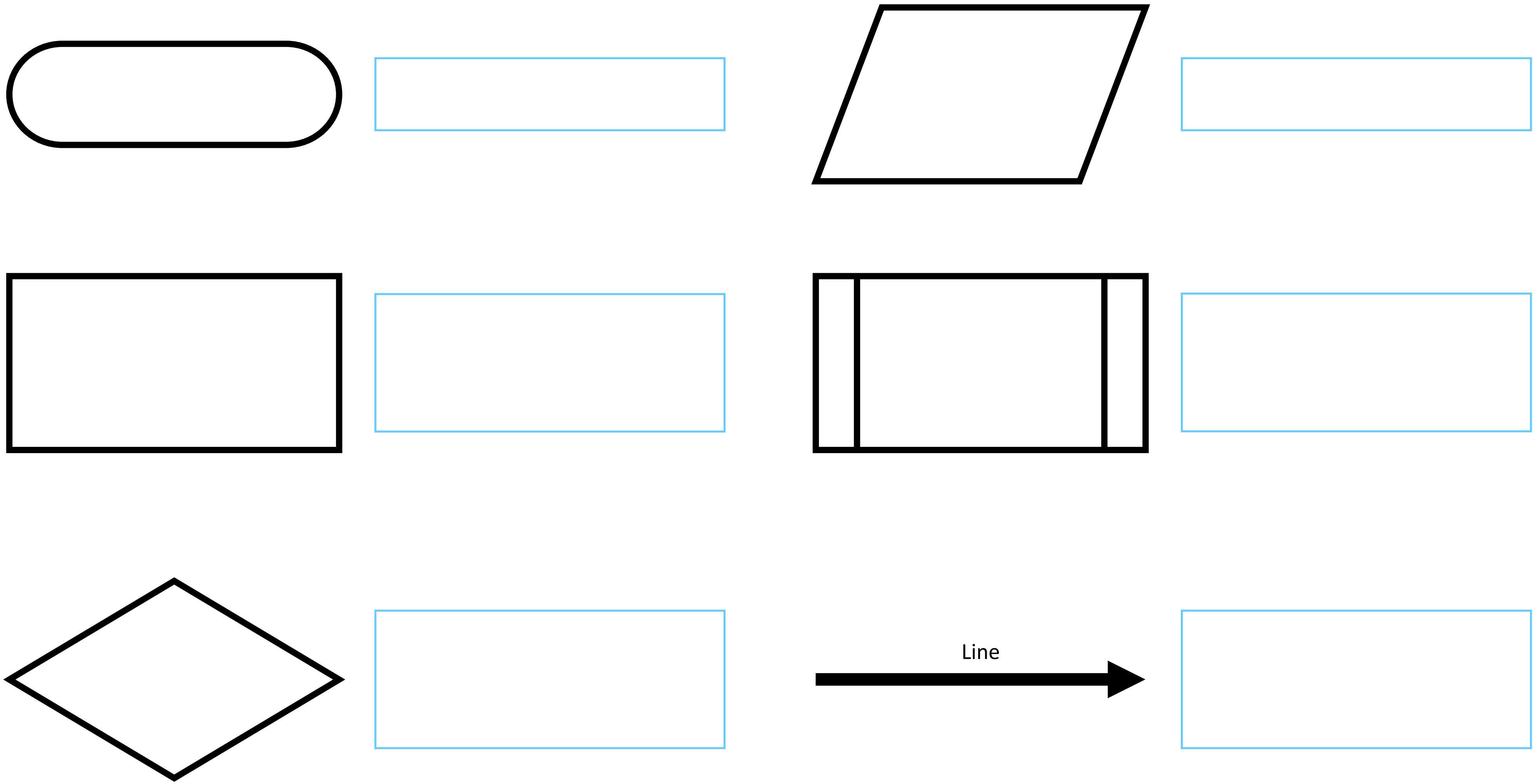
How an insertion sort works:

Yellow dotted box:
unsorted data in the list:

Green solid box:
sorted data in the list:



Flow diagram symbols



How to produce algorithms using flow diagrams

An algorithm for an RPG game displays 3 choices from a menu and allows the user to enter their choice.

1. Play game
2. Change character
3. Quit

The user input is validated so only the numbers 1-3 can be entered.

Interpret, correct or complete algorithms.

An algorithm for an RPG game displays 3 choices from a menu and allows the user to enter their choice.

1. Play game
2. Change character
3. Quit

The user input is validated so only the numbers 1-3 can be entered.

```
DO
    OUTPUT “1. Play game”
    OUTPUT “2. Change character”
    OUTPUT “3. Quit”

    INPUT INT(choice)

WHILE choice<1 AND choice>4
```

How to produce algorithms using flow diagrams

An algorithm for an RPG game handles a battle between two player characters.

Each character has an attack and defence attribute that must be input by the user before an engagement.

When the two characters engage, a random number between 1 and 12 is generated for each player.

The attack attribute plus the defence attribute is added to the player's dice roll.

If player 1's total is greater than player 2's total, player 1 wins otherwise player 2 wins.

The winner is output.

How to produce algorithms using pseudocode

An algorithm for an RPG game handles a battle between two player characters.

Each character has an attack and defence attribute that must be input by the user before an engagement.

When the two characters engage, a random number between 1 and 12 is generated for each player.

The attack attribute plus the defence attribute is added to the player's dice roll.

If player 1's total is greater than player 2's total, player 1 wins otherwise player 2 wins.

The winner is output.

Interpret, correct or complete algorithms.

Modified algorithm to correct an issue with player 2 winning more battles than player 1.

Interpret, correct or complete algorithms.

An algorithm for an RPG game generates a list of random caverns into which objects will be placed.

Caverns are numbered 1-50.
The number of caverns to return is n.

```
FUNCTION randomcaverns(n)
    caverns = []
    FOR c = 1 TO n
        valid = TRUE
        WHILE valid = FALSE
            r = RANDOM (1,50)
            valid = FALSE
            FOR i = 0 TO caverns.LENGTH
                if caverns[i] = r THEN valid = FALSE
            NEXT i
        ENDWHILE
    caverns[c] = c
    NEXT c
    RETURN caverns
ENDFUNCTION
```

How to produce algorithms using flow diagrams

An RPG game allows a player to input their next move by entering N, E, S or W. The valid moves are stored in a list like this: move = [0,1,0,1]
Zero means the move is not possible. One means it is possible. The possibilities are stored in the list in the order: N, E, S, W.

A function takes two parameters:
m is the move: "N", "E", "S" or "W";
vm is a list of the valid moves.

Assuming a zero indexed list/array.

How to produce algorithms using pseudocode

An RPG game allows a player to input their next move by entering N, E, S or W. The valid moves are stored in a list like this: move = [0,1,0,1]
Zero means the move is not possible. One means it is possible. The possibilities are stored in the list in the order: N, E, S, W.

A function takes two parameters:
m is the move: “N”, “E”, “S” or “W”;
vm is a list of the valid moves.

Assessment

Target: [ ] Overall grade: [ ]

Minimum expectations by the end of this unit

☐	You should have learnt terms 100-111 from your GCSE Level Key Terminology during this unit.
☐	You have completed all the pages of the workbook
☐	Score 80% in the end of unit test.

Feedback

Breadth	Depth	Understanding
☐ All aspects complete	☐ Excellent level of depth	☐ All work is accurate
☐ Most aspects complete	☐ Good level of depth	☐ Most work is accurate
☐ Some aspects complete	☐ Basic level of depth shown	☐ Some work is accurate
☐ Little work complete	☐ Little depth and detail provided	☐ Little work is accurate

Comment & action	Student response

Reflection & Revision checklist

Confidence	Clarification
☹ 😐 ☺	I can explain what is meant by the term abstraction.
☹ 😐 ☺	I can explain why abstraction is helpful when we are designing a solution to a problem.
☹ 😐 ☺	I can explain what decomposition is and how it is useful.
☹ 😐 ☺	I can explain what is meant be ‘algorithmic thinking’.
☹ 😐 ☺	I can explain how a binary search works.
☹ 😐 ☺	I can explain how a linear search works.
☹ 😐 ☺	I can explain how a bubble sort works.
☹ 😐 ☺	I can explain how a merge sort works.
☹ 😐 ☺	I can explain how an insertion sort works.
☹ 😐 ☺	I can explain how to produce pseudocode to describe an algorithm and why it is needed.
☹ 😐 ☺	I can explain how to produce a flow diagram to describe an algorithm.
☹ 😐 ☺	I can interpret, correct and complete a range of algorithms.

My revision focus will need to be: