

## ICT KS4 Year 10 Spring 1 Blended Learning Booklet

### Programming

Name:

Form:

- Use BBC Bitesize for any research you may need to carry out to complete this book.
- You can also use the website [www.TeachICT.com](http://www.TeachICT.com)

**Username: cm187nq**

**Password: python4**

- GCSE POD
- YOU TUBE Channel – AQA Tutor

## Contents

Page 3: Big Picture - Year 10 Overview

Page 4: Zoom in - My Learning Journey

Pages 5 and 6 – lesson 1

Pages 7 and 8 lesson 2

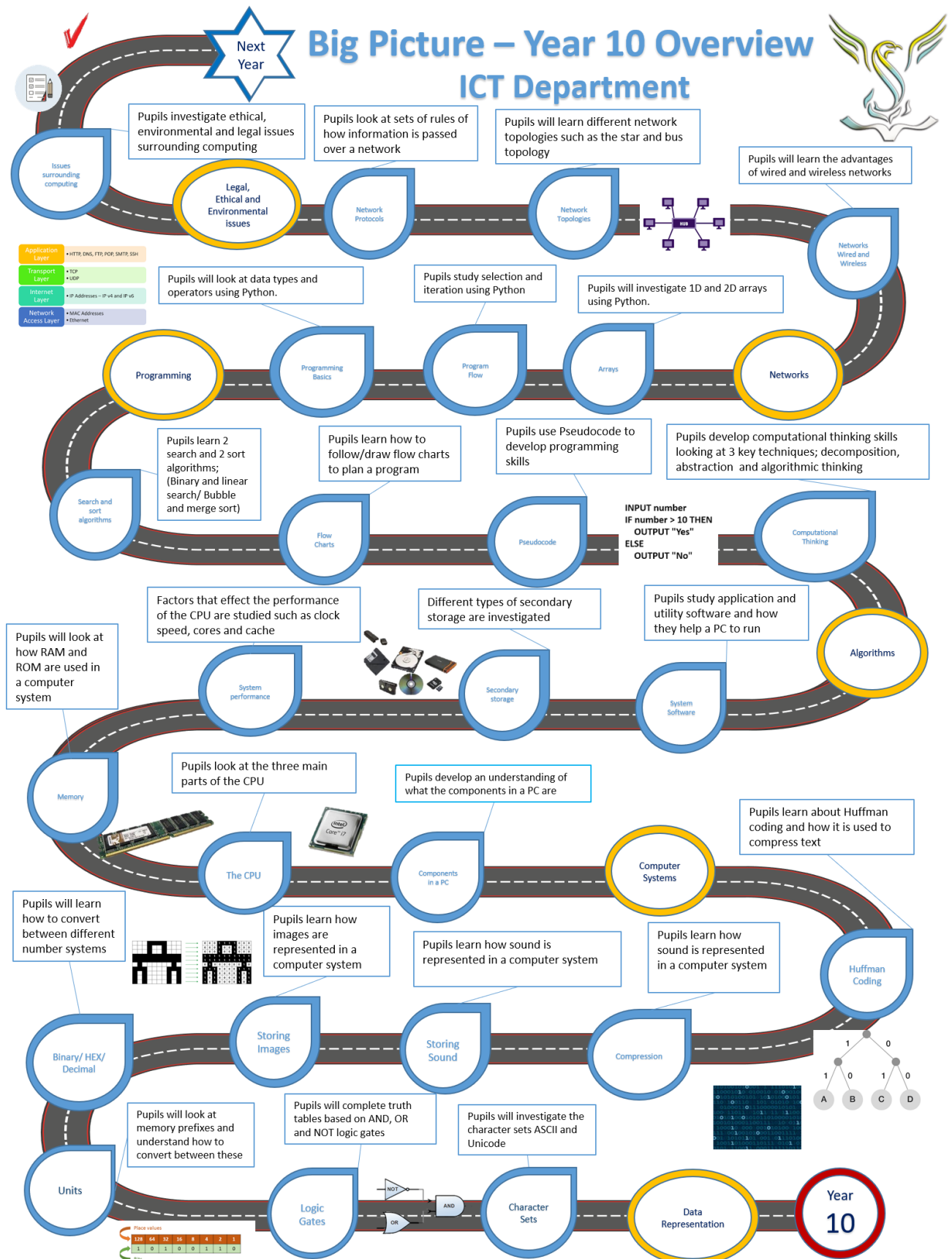
Pages 9, 10 and 11 – lesson 3

Pages 12 and 13 – Lesson 4

Pages 14 – 17 – Lesson 5

Pages 18 – Lesson 6

Pages 19 – Lessons 7 -12





## ZOOM IN...

### MY LEARNING JOURNEY:

*Subject: Computer Science*

*Year: 10 Unit: Algorithms and programming*

#### AIMS

To introduce students to Algorithmic thinking and programming in Python.

#### DEVELOPING COURAGE

- C Writing, speaking and listening to others Using body language to help communication
- O Decision making to find solutions
- U Showing respect for others in the team and valuing their contributions
- R Staying with a problem until it is resolved
- A Monitoring performance and devising strategies for improvement
- G Taking on roles and responsibilities
- E Communicating in a variety of ways, including electronic and social media

#### PREVIOUS LEARNING

- Students studied programming in year 8 and 9 using Python.

#### WHAT WE KNOW/REMEMBER

- .....
- .....
- .....
- .....
- .....

#### UP NEXT

Students will look at networks next term.

We will Look at different types of networks and different protocols that allow computers to communicate with each other.

#### CAREERS

- Technician
- PC Builder
- Computer
- Hardware Engineer

#### PERSONAL OBJECTIVES

- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....
- .....

#### RECOMMENDED READING

- Teach ICT
- GCSE POD
- BBC Bitesize

## **Lesson 1**

LI: to learn about computational thinking; Decomposition, abstraction and algorithmic thinking

### **DART**

What is computational thinking?

Computers can be used to help us solve problems. However, before a problem can be tackled, the problem itself and the ways in which it could be solved need to be understood.

Computational thinking allows us to do this.

### **The four cornerstones of computational thinking**

There are four key techniques (cornerstones) to computational thinking:

- decomposition - breaking down a complex problem or system into smaller, more manageable parts
- pattern recognition – looking for similarities among and within problems
- abstraction – focusing on the important information only, ignoring irrelevant detail
- algorithms - developing a step-by-step solution to the problem, or the rules to follow to solve the problem

Each cornerstone is as important as the others. They are like legs on a table - if one leg is missing, the table will probably collapse. Correctly applying all four techniques will help when programming a computer.

### **Decomposition**

Use the link below to have a go at decomposing some tasks. Write down how you would decompose these tasks. – Make notes in your home learning exercise book.

[https://teach-ict.com/2016/GCSE\\_Computing/AQA\\_8525/3\\_1\\_fundamentals\\_of\\_algorithms/computational\\_thinking/decomposition\\_tasks/tasks.php](https://teach-ict.com/2016/GCSE_Computing/AQA_8525/3_1_fundamentals_of_algorithms/computational_thinking/decomposition_tasks/tasks.php)

Check your answers using the website.

**Abstraction**

Use the link below to have a go at using abstraction. Answer all questions in your home learning exercise book.

[https://teach-ict.com/2016/GCSE\\_Computing/AQA\\_8525/3\\_1\\_fundamentals\\_of\\_algorithms/computational\\_thinking/abstraction\\_tasks/tasks.php](https://teach-ict.com/2016/GCSE_Computing/AQA_8525/3_1_fundamentals_of_algorithms/computational_thinking/abstraction_tasks/tasks.php)

**Algorithmic Thinking**

This is a logical way of getting from the problem to the solution. If the steps you take to solve a problem follow an algorithm then they can be reused and adapted to solve similar problems in the future.

**Questions to answer**

1. What is meant by decomposition?
2. What is meant by abstraction?
3. Why is algorithmic thinking useful when solving a problem?
4. Outline the decomposition, abstraction and algorithmic thinking when choosing a film at the cinema.

## Lesson 2

LI: to learn about pseudocode and how it is used

Algorithms are just sets of instructions for solving a problem. In real-life they can take the forms of recipes, assembly instructions, directions, etc. but in computer science they are often written in pseudo-code.

### Algorithms can be written using **Pseudo-code**

- 1) Pseudo-code is not an actual programming language but it should follow a similar structure and read like one (roughly). The idea is that pseudo-code clearly shows an algorithm's steps without worrying about the finer details (syntax) of any particular programming language.
- 2) It is quick to write and can be easily converted into any programming language.
- 3) There are different ways to write pseudo-code — they are all equally correct as long as the person reading the code can follow it and understand what you mean.

#### EXAMPLE:

Write an algorithm using pseudo-code to calculate the salary of a worker after a 10% pay increase.

A simple solution to the problem would be:

```
Take worker's current salary
Multiply the salary by 1.1
Display the answer
```

This solution is perfectly adequate as the problem has been split down into steps and it is obvious to the reader what to do at each stage.

A more useful solution is shown here:

```
salary ← USERINPUT
newsalary ← salary * 1.1
OUTPUT newsalary
```

This solution is better as the words and structure resemble a real programming language. It can be more easily adapted into real code.

A courier company allows customers to track the delivery of parcels.

To track a specific parcel, customers enter an eight-digit parcel number.

Write some pseudo-code that checks whether a customer has entered a valid parcel number.

The pseudo-code should:

- Ask the user to enter a parcel number.
- Output 'Valid parcel number' if there are eight characters.
- Output 'Not a valid parcel number' if there are not eight characters.

OUTPUT '.....'

parcelNumber ← USERINPUT

IF ..... THEN

ELSE

ENDIF

[Total 4 marks]

## Make sure your pseudo-code isn't *Too Vague*

Even though pseudo-code isn't a formal **programming language** you still need to make sure it's **readable**, **easy to interpret** and not too **vague**.



### EXAMPLE:

When registering on a website, a user's password should be more than 6 characters long and it must be different from their username. Write an algorithm to check if the password is valid. If it's invalid it should say why.

IF the length of the password is less than or equal to 6 characters long OR password is the same as the username THEN it is invalid ELSE the password is valid

This code is **too vague** and **unstructured**. It won't give reasons why the password is invalid and doesn't show the **inputs** and **outputs** (see p12).

The pseudo-code asks the user to **input** a username and password and stores them as **variables**.

The code gives **different outputs** depending on why the password is **invalid**.

Notice that the **indentation** of the pseudo-code makes it more **readable**.

```
username ← USERINPUT
password ← USERINPUT
IF length of password ≤ 6 THEN
    OUTPUT 'Password is too short.'
ELSE
    IF password = username THEN
        OUTPUT 'Password is the same as username.'
    ELSE
        OUTPUT 'Password is valid.'
    ENDIF
ENDIF
```

The first IF statement checks to see if the password is **too short** and the second checks if it's the **same as the username**.

See p12 for more on IF statements.

## Questions

What is pseudocode?

What are three features of well written pseudocode?

What are the benefits of writing algorithms in pseudocode rather than a programming language?

## Extension

Develop an algorithm using pseudocode or a flowchart that calculates an estimate of the braking distance in metres for a new model of go-kart that is travelling between 10 and 50 kilometres per hour (kph).

Your algorithm should be based on the following method:

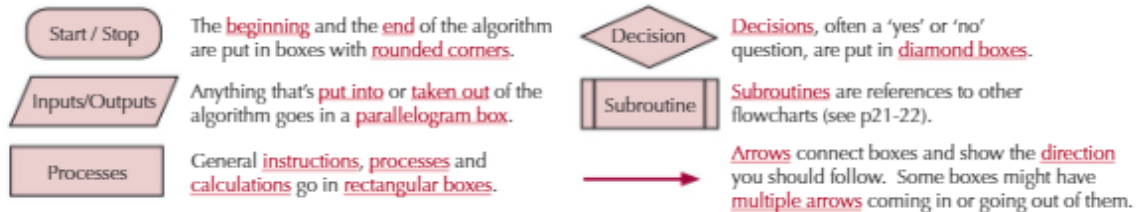
- the user should keep being asked to enter a speed for the go-kart until they enter a speed that is between 10 and 50 (both 10 and 50 are valid speeds)
- the braking distance in metres is calculated by dividing the speed by 5
- the user should be asked if the ground is wet (expect the user to enter 'yes' if it is)
- the braking distance should be multiplied by 1.5 when the ground is wet
- finally, your algorithm should output the calculated braking distance.

## Lesson 3

LI: to learn how to use flowchart when creating an algorithm.

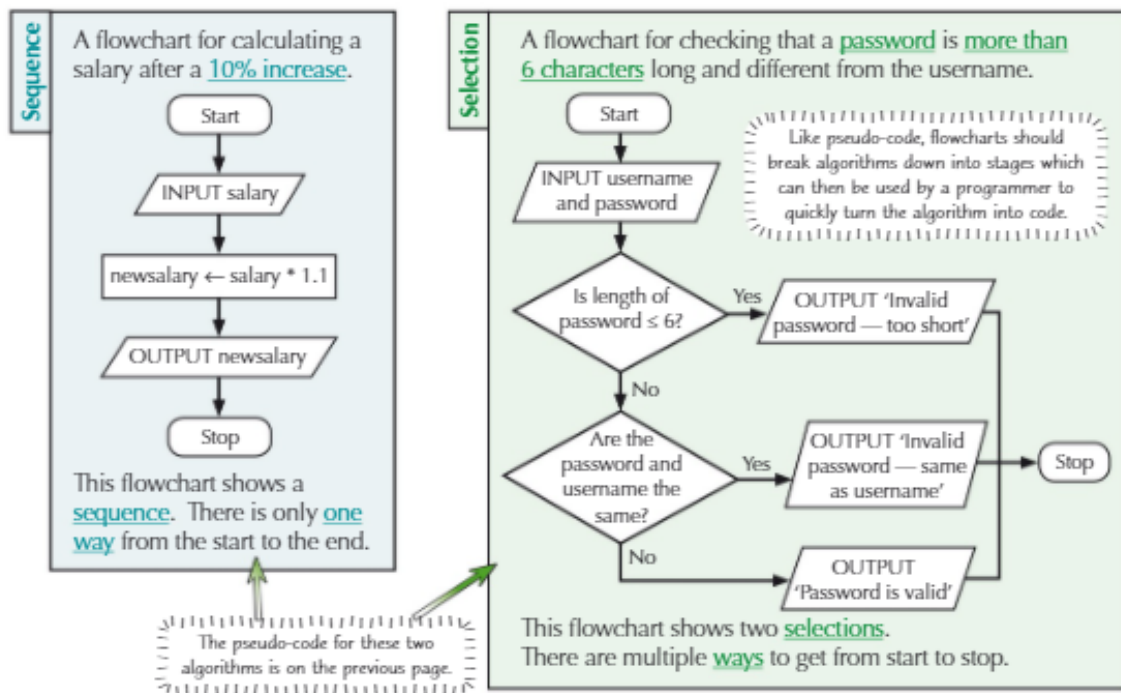
Algorithms can also be shown using a flowchart, and just like for pseudo-code, there are different ways to write the same algorithm. You do get to draw some different shapes though, so things are looking up.

### Flowcharts use **Different Boxes** for different **Commands**



### Algorithms can be written as **Flowcharts**

Flowcharts can show **sequence**, **selection**, **iteration** or a combination of them.



## Question

Draw a flowchart to check if a new username is valid. Usernames should be at least 5 characters long and unique. If it is invalid, the algorithm should give the reason why and get the user to enter another username.



Click on the link below. Read the theory on the BBC Bitesize website and have a go at the test.

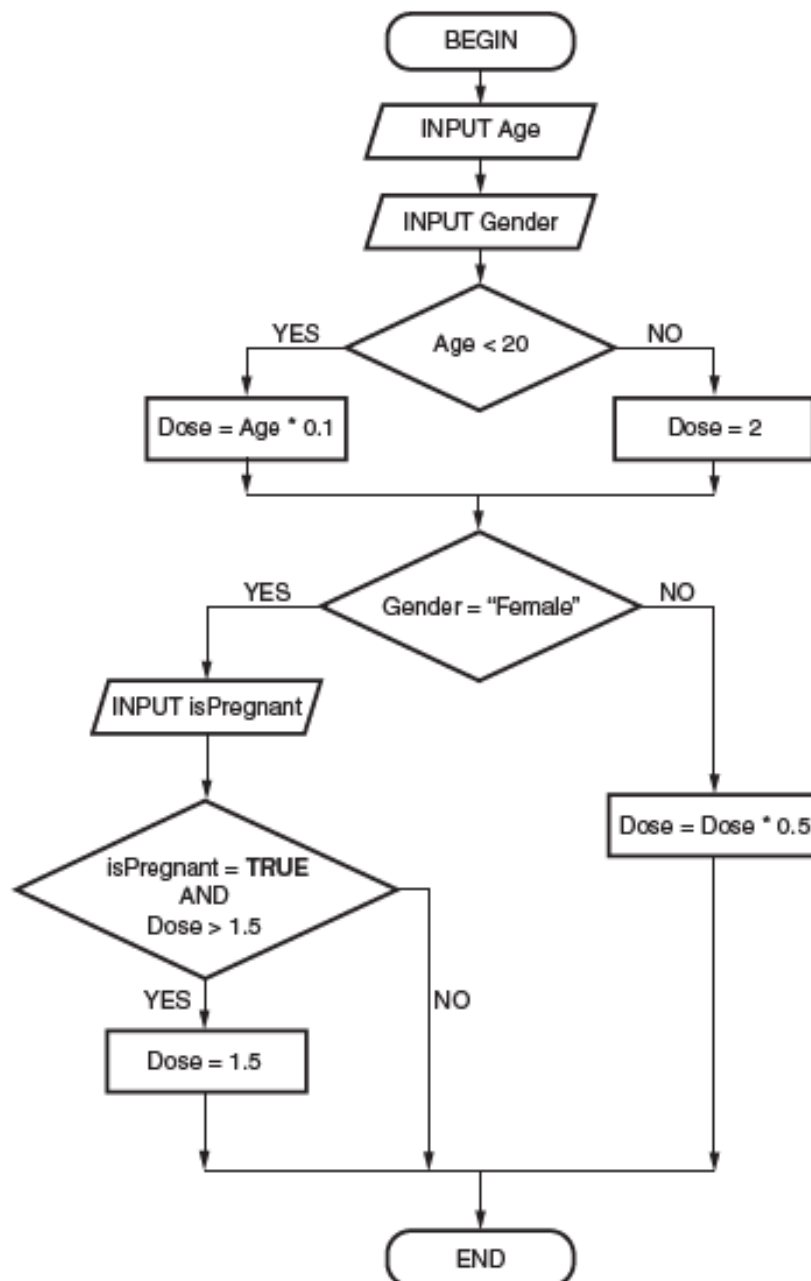
<https://www.bbc.co.uk/bitesize/guides/z3bq7ty/revision/3>

Look at the flow chart below and answer the questions

---

3 A computer program calculates the correct dose in grams of a type of medicine.

The algorithm used is shown by the flow diagram below.





(a) The data type of the variable Age is Integer.

State the data type of the following variables used in the flow diagram.

Variable	Data Type
Gender	
Dose	
isPregnant	

[3]

(b) Use the flow diagram to calculate the correct dose of medicine for a male aged 30.

You must show your working.

[illegible]

(c) Use the flow diagram to calculate the correct dose of medicine for a pregnant female aged 19. You must show your working.

---

---

---

---

---

---

---

---

---

---

## Lesson 4

LI: To learn how the binary and linear search algorithms work

### A Binary Search looks for items in an Ordered List

#### BINARY SEARCH ALGORITHM

- 1) Find the **middle item** in the ordered list.
- 2) If this is the item you're looking for, then **stop** the search — you've found it.
- 3) If not, **compare** the item you're **looking for** to the **middle item**. If it comes **before** the middle item, get rid of the **second half** of the list. If it comes **after** the middle item, get rid of the **first half** of the list.
- 4) You'll be left with a list that is **half the size** of the original list. Repeat steps 1) – 3) on this **smaller list** to get an even smaller one. Keep going until you find the item you're looking for.

To find the **middle item** in a list of  $n$  items do  $(n + 1) \div 2$  and round up if necessary.

OK, so get rid of all the teeth on the left...



#### EXAMPLE:

Use the binary search algorithm to find the number 99 in the following list.

7	21	52	59	68	92	94	99	133
---	----	----	----	----	----	----	----	-----

There are 9 items in the list so the middle item is the  $(9 + 1) \div 2 = 5$ th item.

The 5th item is 68 and  $68 < 99$  so get rid of the **first half** of the list to leave:

92	94	99	133
----	----	----	-----

There are 4 items left so the middle item is the  $(4 + 1) \div 2 = 2.5 = 3$ rd item

The 3rd item is 99. You've found the item you're looking for so the search is complete.

## Question

Sonia has a sorted list of ice cream flavours that she sells in her shop.

- a) Show the stages of a binary search to find the word 'butterscotch' in the list below.

butterscotch	chocolate	mint	strawberry	vanilla
--------------	-----------	------	------------	---------

[4]

- b) Give **one** advantage of using a binary search over a linear search.

.....

.....

[1]

- c) Six more flavours are added to Sonia's list. The list is still in alphabetical order. Explain why it would only take two iterations of a binary search to find the ninth item in the list.

.....

.....

.....

.....

[3]

## Linear Search

### A Linear Search can be used on an Unordered List

A linear search checks each item of the list in turn to see if it's the correct one. It stops when it either finds the item it's looking for, or has checked every item.

It's important that you show every step when following these algorithms.

#### LINEAR SEARCH ALGORITHM

- 1) Look at the first item in the unordered list.
- 2) If this is the item you're looking for, then stop the search — you've found it.
- 3) If not, then look at the next item in the list.
- 4) Repeat steps 2) – 3) until you find the item that you're looking for or you've checked every item.

- 1) A linear search is much simpler than a binary search but not as efficient (see p32). The biggest advantage of a linear search is that it can be used on any type of list, it doesn't have to be ordered.
- 2) For small ordered lists the difference in efficiency doesn't really matter so the run time of both algorithms will be similar.
- 3) For large ordered lists the run time of binary search will generally be much quicker than linear search.

#### EXAMPLE:

Use a linear search to find the number 99 from the list above.

Check the first item:  $7 \neq 99$

Look at the next item:  $21 \neq 99$

Look at the next item:  $52 \neq 99$

Look at the next item:  $59 \neq 99$

Look at the next item:  $68 \neq 99$

Look at the next item:  $92 \neq 99$

Look at the next item:  $94 \neq 99$

Look at the next item:  $99 = 99$

You've found the item you're looking for so the search is complete.

A programmer wants to implement a search algorithm to be used with small arrays. The figure below shows an example array.

[4, 6, 8, 12, 15, 16, 21]

- (a) Using the figure above, explain how linear search would search for the integer 15.

---

---

---

---

---

---

---

---

---

## Lesson 5

LI: To learn how the bubble sort and merge sort algorithms work.

### A Bubble Sort compares Pairs of items

The **bubble sort algorithm** is used to sort an unordered list of items.

The algorithm is **very simple** to follow but can often take a while to actually sort a list.

BUBBLE SORT ALGORITHM	1) Look at the <b>first two items</b> in the list.
	2) If they're in the <b>right order</b> , you don't have to do anything. If they're in the <b>wrong order</b> , <b>swap them</b> .
	3) Move on to the <b>next pair</b> of items (the 2nd and 3rd entries) and repeat step 2).
	4) Repeat step 3) until you get to the <b>end</b> of the list — this is called one <b>pass</b> . The <b>last item</b> will now be in the correct place, so <b>don't include</b> it in the next pass.
	5) Repeat steps 1) – 4) until there are <b>no swaps</b> in a pass.

Start from the beginning.  
Each pass will have one less  
comparison than the one before it.  
Keep repeating until sorted.

#### EXAMPLE:

Use the bubble sort algorithm to write these numbers in ascending order.

66 21 38 15 89 49

##### First pass

66	21	38	15	89	49	Compare 66 and 21 — swap them.
21	66	38	15	89	49	Compare 66 and 38 — swap them.
21	38	66	15	89	49	Compare 66 and 15 — swap them.
21	38	15	66	89	49	Compare 66 and 89 — no swap.
21	38	15	66	89	49	Compare 89 and 49 — swap them.
21	38	15	66	49	89	End of first pass.

After the **2nd pass** the order of the numbers will be: 21 15 38 49 66 89

After the **3rd pass** the order of the numbers will be: 15 21 38 49 66 89

There are **no swaps** in the 4th pass so the list has been sorted: 15 21 38 49 66 89

The bubble sort is considered to be one of the simplest sorting algorithms as it only ever focuses on **two items** rather than the whole list of items.

Pros	It's a <b>simple algorithm</b> that can be easily implemented on a computer.
	It's an <b>efficient way</b> to <b>check</b> if a list is <b>already in order</b> . For a list of $n$ items you only have to do one pass of $n - 1$ comparisons to check if the list is ordered or not.
	Doesn't use very much <b>memory</b> as all the sorting is done using the <b>original list</b> .

Cons	It's an <b>inefficient way</b> to <b>sort a list</b> — for a list of $n$ items, the <b>worst case scenario</b> would involve you doing $\frac{n(n-1)}{2}$ comparisons.
	Due to being <b>inefficient</b> , the bubble sort algorithm is pretty <b>slow</b> for <b>very large lists</b> of items.

Once a list is sorted, there's no need to keep comparing it.  
Already sorted, no need to keep comparing it.  
Already sorted, no need to keep comparing it.  
Already sorted, no need to keep comparing it.

But then, the items are already sorted.

Enjoy watching how the bubble sort works in dance form!

<https://www.youtube.com/watch?v=lyZQPjUT5B4>



Six athletes compete in a long jump competition. Their best jumps are shown below.

5.32 m	5.50 m	5.39 m	6.50 m	6.28 m	6.14 m
--------	--------	--------	--------	--------	--------

- a) Show the stages of a bubble sort to list these jumps in order from shortest to longest.

[4]

Fill in the blank arrays to show the steps involved in applying the bubble sort algorithm to the array [3, 5, 1, 4, 2]. You only need to show the missing steps where a change is applied to the array.

3	5	1	4	2
1	2	3	4	5

## A Merge Sort Splits the list apart then Merges it back together

The merge sort algorithm is an example of a **divide-and-conquer** algorithm and takes advantage of two facts:

- Small lists are **easier to sort** than large lists.
- It's easier to merge **two ordered lists** than two unordered lists.

### MERGE SORT ALGORITHM

- 1) **Split** the list in **half** (the smaller lists are called **sub-lists**) — the second sub-list should start at the **middle item** (see p4).
- 2) Keep repeating step 1) on each sub-list until **all the lists only contain one item**.
- 3) **Merge** pairs of sub-lists so that each sub-list has twice as many items. Each time you merge sub-lists, **sort the items** into the right order.
- 4) Repeat step 3) until you've merged **all the sub-lists** together.

### EXAMPLE

Use the merge sort algorithm to write these letters in alphabetical order.

- 1) **Split** the original list of 8 items into **two lists**, the second list should start at the  $(8 + 1) \div 2 = 4.5 \rightarrow$  **5th item**.
- 2) Carry on **splitting** the sub-lists until each list only has **one item** in it.
- 3) **Merge and order** sub-lists back together. Note that merging is always performed on **two ordered lists** and is **very simple** to do. Eg.  
Compare F and A — move A to the new list.  
Compare F and L — move F to the new list.  
Compare P and L — move L to the new list.  
P is the last item in the new list.
- 4) Keep **merging** sub-lists until you only have **one list**.



You'll often be unable to **split** or **merge** lists **easily**. For example, sometimes you'll have to merge a list containing **two items** with a list containing **one item** to make a list of **three items**.

Pros	Cons
<ul style="list-style-type: none"> <li>• In general, it's much <b>more efficient</b> and <b>quicker</b> than the bubble sort algorithm (p5) for large lists, and has a <b>similar running time</b> for short lists.</li> <li>• It has a very <b>consistent running time</b> regardless of how ordered the items in the original list are.</li> </ul>	<ul style="list-style-type: none"> <li>• Even if the list is <b>already sorted</b>, it still goes through the whole <b>splitting</b> and <b>merging</b> process, so a bubble sort may be <b>quicker</b> in some cases.</li> <li>• It uses <b>more memory</b> than the bubble sort because it has to create additional lists.</li> </ul>

Due to its efficiency, the merge sort algorithm (or variations of it) are used in many programming languages such as Java®, Python® and Perl as the primary sorting algorithm.

Enjoy watching how merge sort works in dance form!

[https://www.youtube.com/watch?v=XaqR3G\\_NVoo](https://www.youtube.com/watch?v=XaqR3G_NVoo)

The list below shows the calorie content of different packets of sweets.

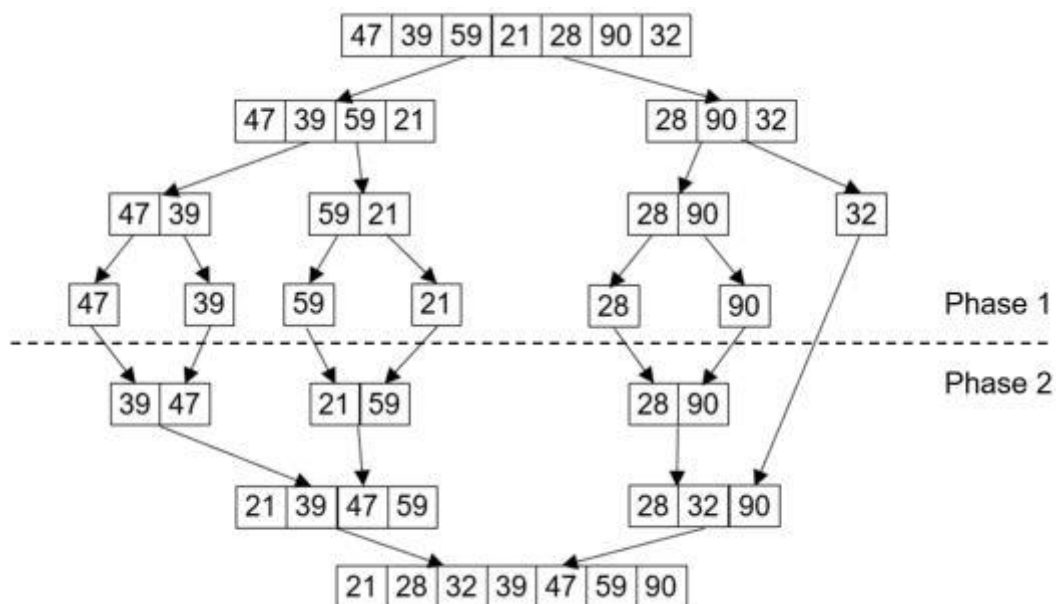
450	350	230	180	650	500	340	270
-----	-----	-----	-----	-----	-----	-----	-----

a) Show the stages of a merge sort to arrange the list from lowest to highest.

[4]

The image below shows a merge sort being carried out on a list.

**Merge sort being carried out on list**



The sort is carried out in two phases. The phases are separated by the dotted line.

(a) Explain the process that is being carried out in **Phase 1** of the sort (above the dotted line).

(3)

(b) Explain the process that is being carried out in **Phase 2** of the sort (below the dotted line).

(3)



## Lesson 6

### Revision Lesson

Please answer the questions below in your home learning exercise book.

#### Search Algorithms (p4) ☐

9) What are the four steps of the binary search algorithm? ☐

10) What are the four steps of a linear search algorithm? ☐

11)\* Here's a fascinating list of British towns and cities:

Ashington, Brecon, Chester, Dagenham, Morpeth, Usk, Watford

a) Use a binary search to find "Morpeth" in this list: ☐

b) Now do the same using a linear search. ☐

12) What are the benefits and drawbacks of using a linear search over a binary search? ☐

#### Sorting Algorithms (p5-6) ☐

13) a) What are the five steps of the bubble sort algorithm?

b)\* Use the bubble sort algorithm to sort these fruit into alphabetical order:

Orange, Banana, Apple, Peach, Grape, Lime ☐

14) What are the four steps of the merge sort algorithm? ☐

15)\* Here is a list of numbers:

8, 7, 5, 1, 3, 6, 4, 2

a) Use the merge sort algorithm to sort this list into ascending order.

b) Use the bubble sort algorithm to sort this list into descending order. ☐

16) Outline the strengths and weaknesses of the following sorting algorithms:

a) bubble sort

b) merge sort ☐




Please use BBC Bitesize or teach ICT to revise around the topics we have studied in the first 6 lessons.

### **Lessons 7 – 12**

LI: To learn about data types, sequence, selection and iteration.

Please click on the link below

<https://classroom.thenational.academy/subjects-by-key-stage/key-stage-4/subjects/computing>

 Computer Science <b>Programming 1: Sequence</b> 5 Lessons	 <b>Programming 2: Selection</b> 6 Lessons	 <b>Programming 3: Iteration</b> 5 Lessons
--	--	---

I would like you to complete the three sections above. Each section has several lessons however these lessons are **not** 75 minutes.

Please work your way through each lesson answering the questions/ worksheets along the way.