2025

COMPUTER SCIENCE

DR. SAW MYAT SANDAR

Pearson Edexcel International GCSEs (O Level)

Unit - 1

CHAPTER 1 - SORTING AND SEARCHING ALGORITHMS

Algorithm

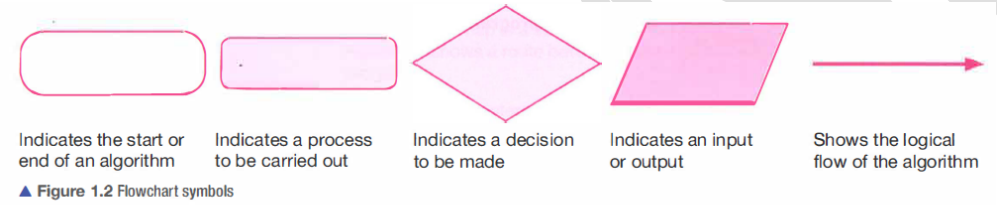
- Step by step instructions/procedures
- Unambiguous
- More than one way

Successful Algorithm

There are **three points** to consider when deciding whether an algorithm is successful or not.
Accuracy - it must lead to the expected outcome (e.g. create a route from Beijing to Shanghai).
Consistency - it must produce the same result each time it is run.
Efficiency - it must solve the problem in the shortest possible time, using as few computer resources as possible.

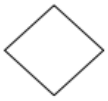
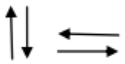
*** ***Algorithms and programs are closely related, but they are not the same.***

Flowchart



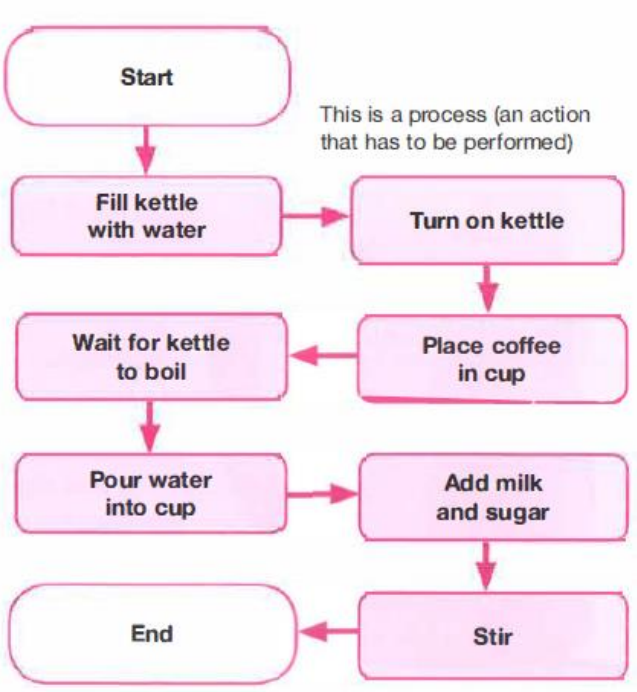
▲ Figure 1.2 Flowchart symbols

The flowchart in Figure 1.3 is an alternative way of showing the algorithm for making a cup of coffee as a written description.

Symbol	Name	Function
	**Process**	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

Algorithm for making a cup of coffee

Fill kettle with water.
Turn on kettle.
Place coffee in cup.
Wait for water to boil.
Pour water into cup.
Add milk and sugar.
Stir.



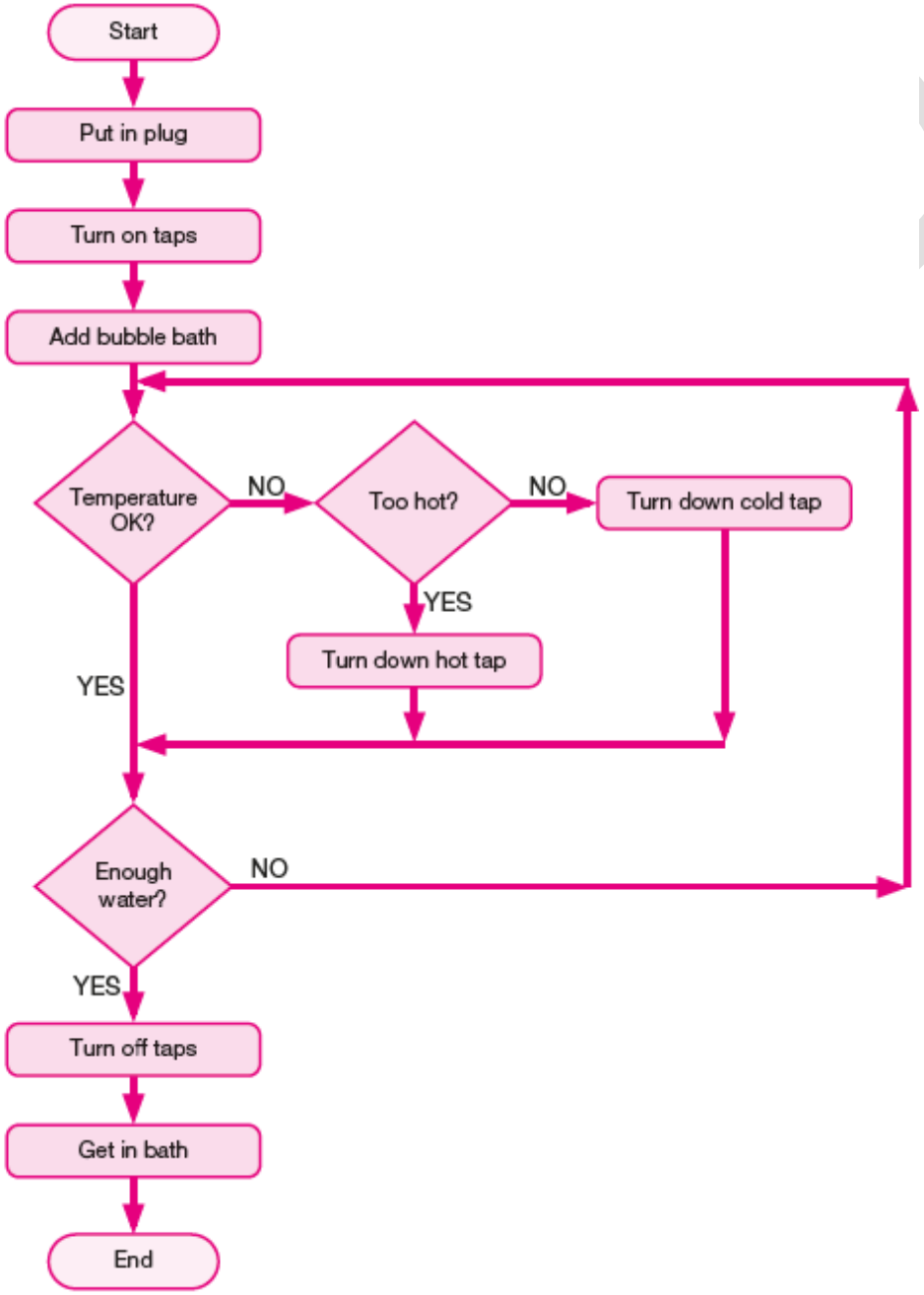
Activity 3

ACTIVITY 3

BATH FLOWCHART

A student has created a written algorithm for preparing a bath. Working with a partner, display the following as a flowchart. You may need to change the order or add actions.

- Put in the plug.
- Fill the bath to the correct level.
- Check the temperature is OK.



Algorithm for adding two numbers

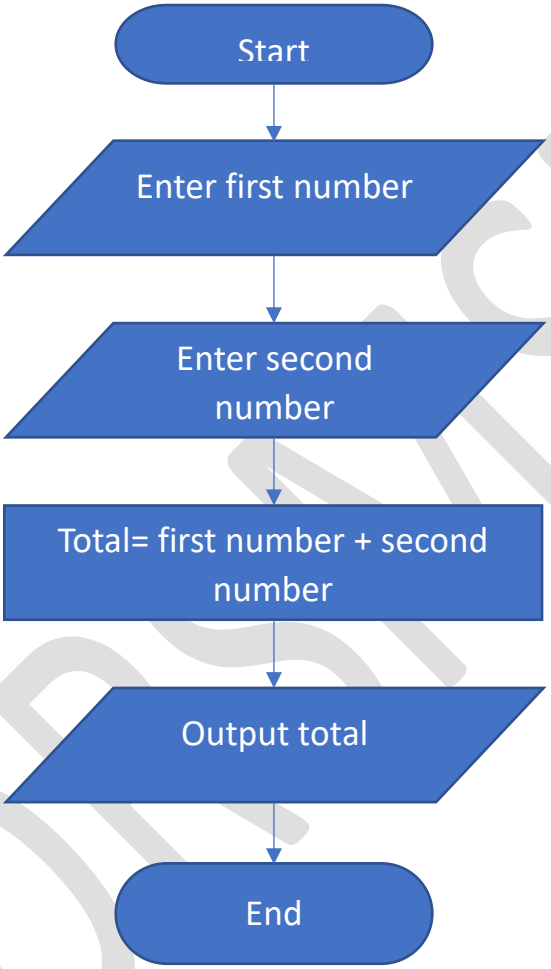
Enter First number.

Enter second number.

Calculate total by adding first and second numbers.

Output total

Flowchart



Pseudocode

```
SEND “Enter first number” TO DISPLAY
RECEIVE firstNumber FROM KEYBOARD
SEND “Enter second number” TO DISPLAY
RECEIVE secondNumber FROM KEYBOARD
SET total TO firstNumber+secondNumber
SEND total TO DISPLAY
```

*** ***Text has to be placed in quotation marks (single or double) if it is to be displayed.*** For example, 'Please enter the first number' (or "Please enter the first number"). ***Quotation marks are not used if a variable is to be displayed.***

Seven Arithmetic Operator

ARITHMETIC OPERATORS

OPERATOR	FUNCTION	EXAMPLE
+	Addition: add the values together.	8 + 5 = 13 myScore1 + myScore2
–	Subtraction: subtract the second value from the first.	17 – 4 = 13 myScore1 – myScore2
*	Multiplication: multiply the values together.	6 * 9 = 54 numberBought * price
/	Real division: divide the first value by the second value and return the result including decimal places.	13 / 4 = 3.25 totalMarks/numberTests
DIV	**Quotient**: like division, but it only returns the whole number or *integer*.	13 DIV 4 = 3 totalMarks DIV numberTests
MOD	**Modulus**/modulo: this will return the remainder of a division.	13 / 4 = 3 remainder 1 Therefore 13 MOD4 = 1
^	Exponentiation: this is for 'to the **power** of'.	3 ^ 3 = 27 It is the same as writing 3^3

▲ Table 1.1 Arithmetic operators

Variables and Constant

Variables play an important role in algorithms and programming.

The value stored by a variable can change as a program is running.

Variables are extremely useful in programming because they make it possible for the same program to process different sets of data.

A constant is the opposite of a variable.

It is a 'container' that holds a value that always stays the same. (eg. PI=3.142)

Constants are useful for storing fixed information, such as the value of pi, the number of litres in a gallon or the number of months in a year.

Each variable and constant in an algorithm has to have a unique identifier.

Activity5

ACTIVITY 5

WRITING ALGORITHMS IN PSEUDOCODE

Here is a written description of an algorithm:

Enter the first number.
Enter the second number.
The third number is equal to the first number multiplied by the second number.
Display the third number.

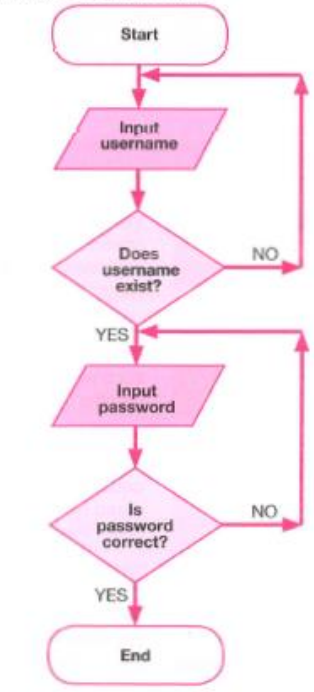
Express this algorithm in pseudocode.

```
1. SEND "Enter the first number" TO DISPLAY
2. RECEIVE firstNumber FROM KEYBOARD
3. SEND "Enter the second number" TO DISPLAY
4. RECEIVE secondNumber FROM KEYBOARD
5. SET thirdNumber TO firstNumber* secondNumber
6. SEND thirdNumber TO DISPLAY
```

Activity 6

WRITTEN DESCRIPTIONS OF ALGORITHMS

This algorithm is displayed as a flowchart.



▲ Figure 1.5 Flowchart of an algorithm

Produce a written description of this algorithm.

```
Ask the user to enter their username.
Repeat until an existing username is entered.
Next ask the user to enter their password.
Repeat until the correct password is entered.
```

CHECKPOINT

S4 What is the difference between a variable and a constant?

```
Variables and constants are 'containers' for storing data.
The value stored in a variable can change,
whereas the value of a constant never changes.
```

C2 Write an algorithm expressed in pseudocode that receives three numbers from the keyboard, then calculates and displays the average.

```
SEND ‘Enter first number.’ TO DISPLAY
RECEIVE firstNumb FROM KEYBOARD
SEND ‘Enter second number.’ TO DISPLAY
RECEIVE secondNumb FROM KEYBOARD
SEND ‘Enter third number.’ TO DISPLAY
RECEIVE thirdNumb FROM KEYBOARD
SET average TO (firstNumb + secondNumb + thirdNumb)/3
SEND average TO DISPLAY
```

CHAPTER 2 - SORTING AND SEARCHING ALGORITHMS

Activity 7

GUESSING GAMES

A student is creating a guessing game. A player has to enter a number no greater than 10. If it is too high, they are informed that they have made an error. But if it is within the range 1 to 10, they are told whether or not they have guessed the correct number. (Assume that the correct number is 3.)

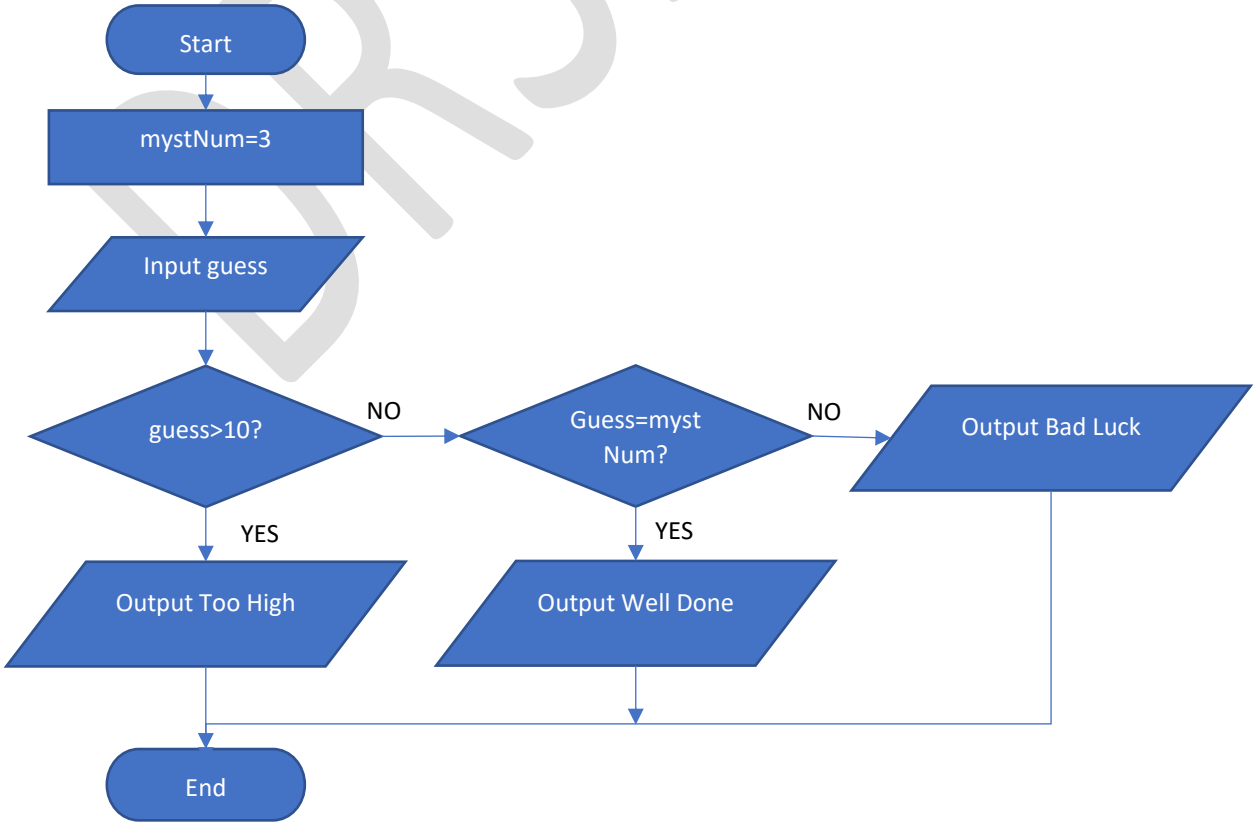
Can you make an algorithm to solve this problem and express it as a written description and a flowchart?

Compare your solution to others in your group.

Check that they are correct and would produce the correct outcome.

Are some of the algorithms more efficient than others? Do they use fewer commands?

```
SET mystNum TO 3
SEND "Enter a number no greater than 10" TO DISPLAY
RECEIVE guess FROM KEYBOARD
IF guess> 10 THEN
      SEND "Too High" TO DISPLAY
ELSE
      IF guess=mystNum THEN
            SEND "Correct" TO DISPLAY
      ELSE
            SEND "Incorrect, Try Again." TO DISPLAY
      END IF
END IF
```



Activity 8

ACTIVITY 8

CALCULATING GRADES

A school uses this algorithm to calculate the grade that students achieve in end-of-topic tests.

```
RECEIVE testScore FROM KEYBOARD
IF testScore >= 80 THEN
    SEND 'A' TO DISPLAY
ELSE
    IF testScore >= 70 THEN
        SEND 'B' TO DISPLAY
    ELSE
        IF testScore >= 60 THEN
            SEND 'C' TO DISPLAY
        ELSE
            IF testScore > 0 THEN
                SEND 'D' TO DISPLAY
            ELSE
                SEND 'FAIL' TO DISPLAY
            END IF
        END IF
    END IF
END IF
```

What would be the output of this algorithm for these test scores: 91, 56 and 78?

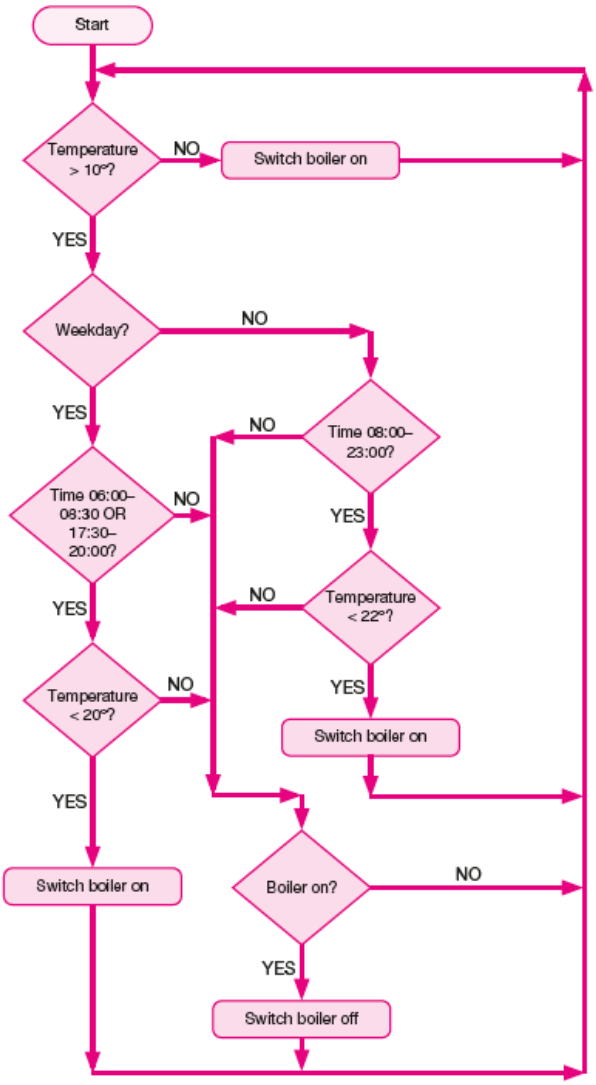
Score	Output
91	**A**
56	**D**
78	**B**
23	D
66	C
0	FAIL

CHECKPOINT

C1 Develop an algorithm using a flowchart that asks the user to enter their height (in metres) and weight (in kilograms) and displays their body mass index (BMI). The formula for calculating BMI is weight/height2.

```
SEND “Enter your height (in metres) ” TO DISPLAY

RECEIVE height FROM KEYBOARD

SEND “Enter your weight (in kilograms) ” TO DISPLAY

RECEIVE weight FROM KEYBOARD

SEND “Your BMI is: ” & weight/height^2 TO DISPLAY
```

C2 Develop an algorithm expressed as a flowchart to control the heating in a house. A thermostat monitors the temperature within the house. During the week the temperature should be 20°C between 06.00 and 08.30 in the morning and between 17.30 and 22.00 at night. At weekends it should be 22°C between 08.00 and 23.00. If the temperature in the house falls below 10°C at any time the boiler is switched on.



CHAPTER 3 - SORTING AND SEARCHING ALGORITHMS

BUBBLE SORT ALGORITHM

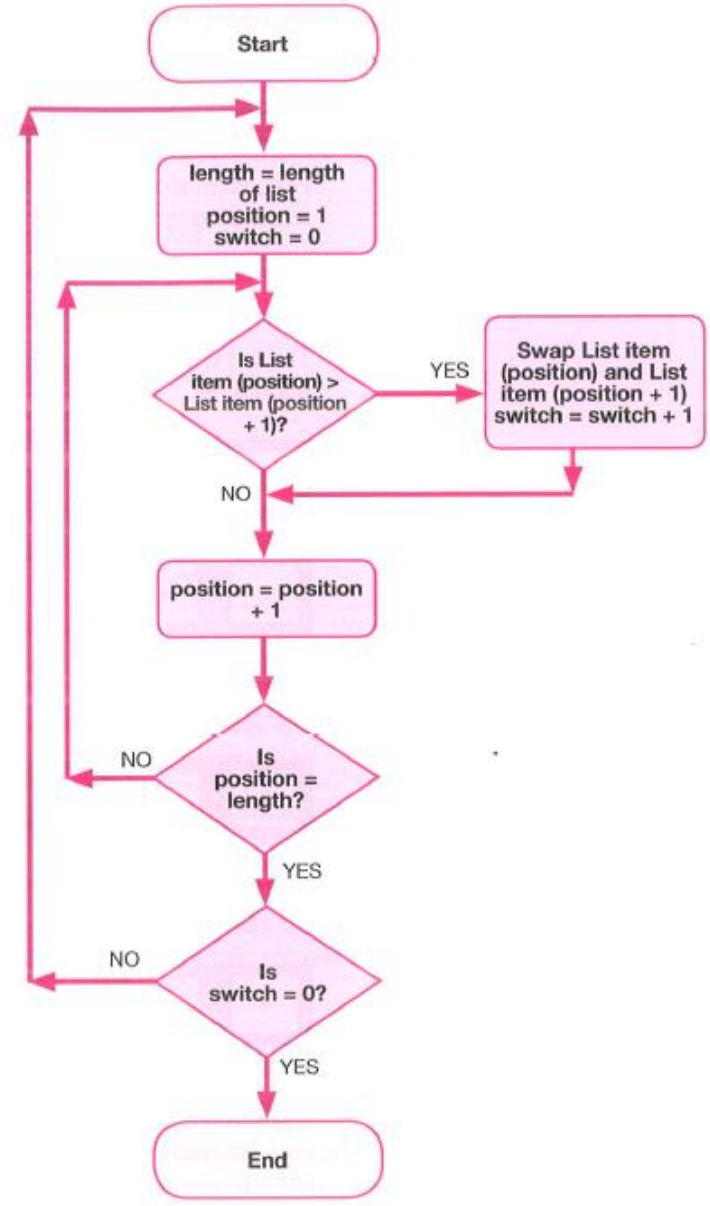
1. Start at the begin of the list
2. Compare the value in position 1 and position 2, if not ascending, swap
3. Compare the value in position 2 and position 3, swap if necessary
4. Continue this process at the end of the list.
5. If swap, repeated step 2 to 4

Activity 9

HOW DOES BUBBLE SORT WORK?

Study the flowchart of the bubble sort algorithm.

Using the variables declared, can you explain the logic behind the algorithm? How does it function to sort a list?



1. The variable length is used to store the length of the list.
2. The variable switch is initially set to 0. Starting at the beginning of the list, successive pairs of items are compared and swapped round if the first item is bigger than the second item.
3. When a swap occurs the value of switch changes from 0 to 1. This process is repeated until the end of the list is reached.
4. If at the end of a pass through the list the value of switch hasn't changed from 0 to 1, this indicates that no swaps have taken place, meaning that the list is now sorted.
5. The algorithm then terminates.

Bubble Sort Worked Example

Pass 1

4 2 6 1 3 – swap

2 4 6 1 3 – no swap

2 4 6 1 3 – swap

2 4 1 6 3 – swap

2 4 1 3 6

The result of first pass - 2 4 1 3 6

How many swaps in first pass – 3

How many comparison in first pass – 4

Pass 2

2 4 1 3 6 - no swap

2 4 1 3 6 – swap

2 1 4 3 6 – swap

2 1 3 4 6 – no swap

2 1 3 4 6

The result of second pass - 2 1 3 4 6

How many swaps in second pass – 2

How many comparison in first pass – 4

Pass 3

2 1 3 4 6 – swap

1 2 3 4 6 – ns

1 2 3 4 6 – ns

1 2 3 4 6 – ns

1 2 3 4 6

The result of second pass – 1 2 3 4 6

How many swaps in second pass – 1

How many comparison in first pass – 4

Pass 4 (think only for computer program)

1 2 3 4 6 - ns

1 2 3 4 6 - ns

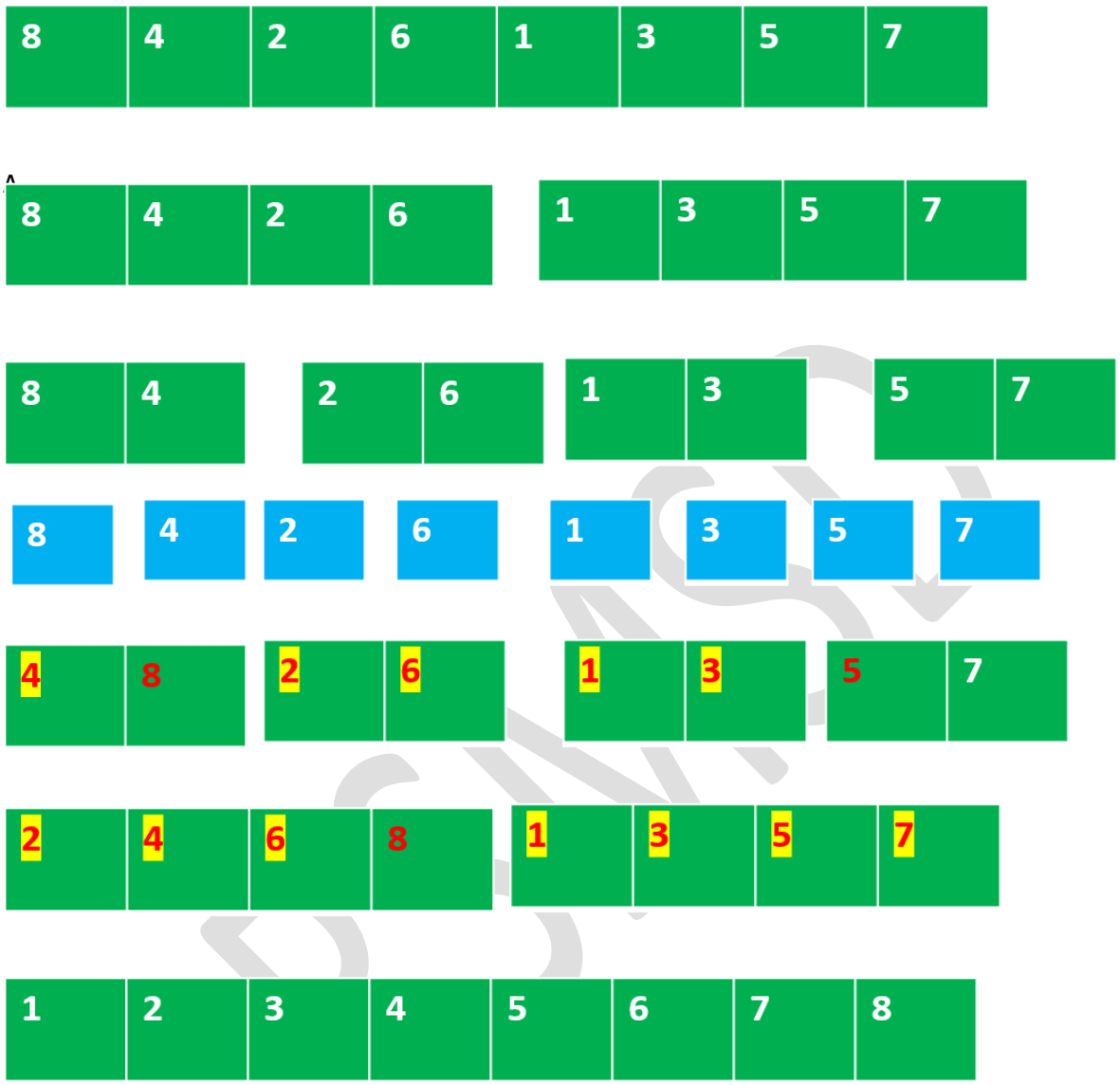
1 2 3 4 6 - ns

1 2 3 4 6 – ns

There is no swap, it is sorted.

MERGE SORT

WORKED EXAMPLE



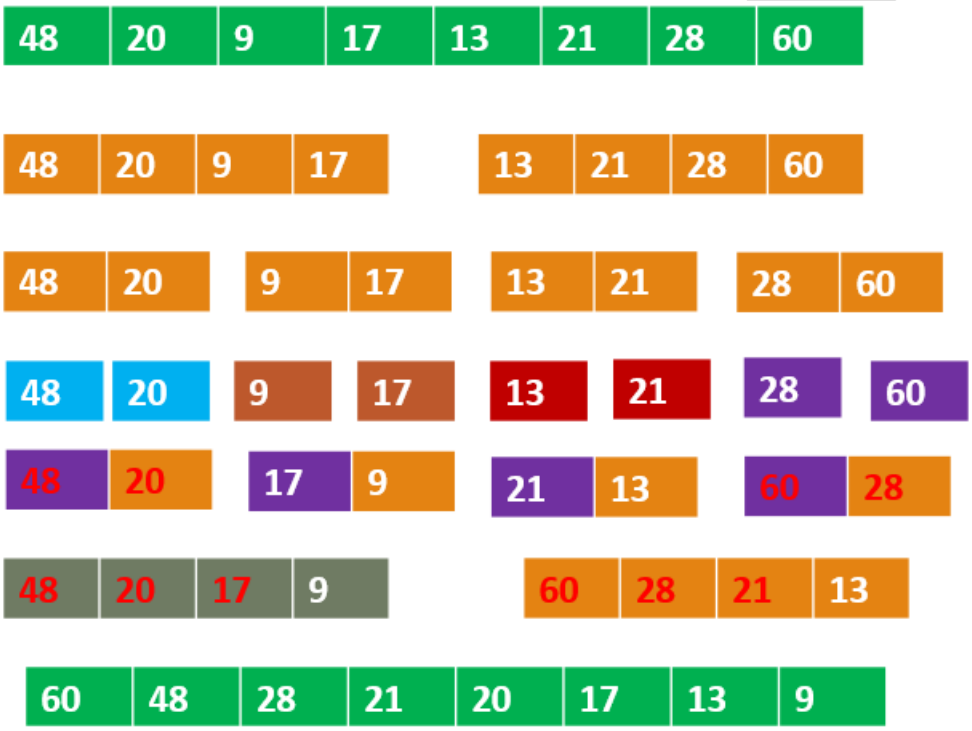
Activity 10

ACTIVITY 10

USING MERGE SORT

Using a table like the one in the worked example on page 18, show how the following list would be sorted into descending order using merge sort.

48, 20, 9, 17, 13, 21, 28, 60



EFFICIENCY OF SORTING ALGORITHMS

Only two sorting algorithms are required for the specification: bubble sort (the slowest) and merge sort (one of the most efficient). There are far more, and many of them are relatively easy to code. Research the insertion and selection sorts.

The **bubble sort algorithm** is said to be using brute force because it starts at the beginning and completes the same task over and over again until it has found a solution.

The **merge sort** uses the divide and conquer method because it repeatedly breaks down the problem into smaller sub-problems, solves those and then combines the solutions.

The graph shows that a bubble sort is far slower at sorting lists of more than 1000 items, but for smaller lists the time difference is too small to be of importance.

LINEAER SEARCH

LINEAR SEARCH

1. Start at the first item in the list.
2. Compare the item with the search item.
3. If they are the same, then stop.
4. If they are not, then move to the next item.
5. Repeat 2 to 4 until the end of the list is reached.

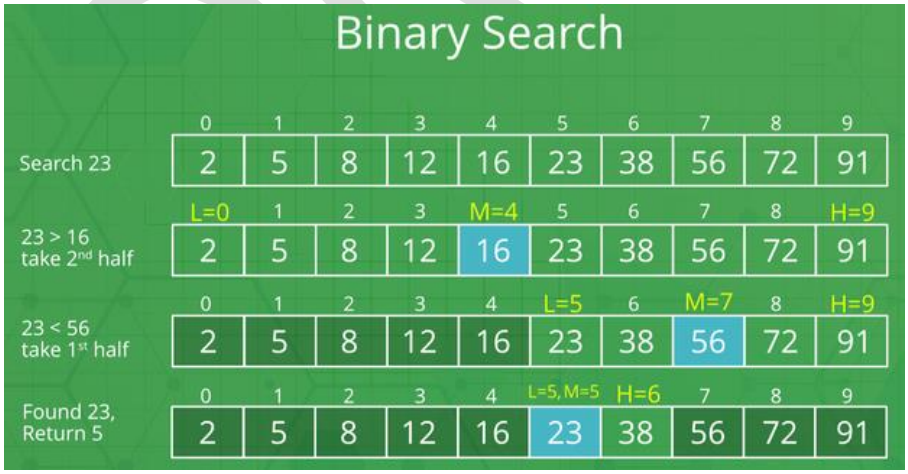
BINARY SEARCH

BINARY SEARCH (ITEMS IN ASCENDING ORDER)

1. Select the median item of the list.
2. If the median item is equal to the search item, then stop.
3. If the median is too high, then repeat 1 and 2 with the sub-list to the left.
4. If the median is too low, then repeat 1 and 2 with the sub-list to the right.
5. Repeat steps 3 and 4 until the item has been found or all of the items have been checked.

Binary Search

1. Select medium
2. M - Equal, stop
3. M - high, left
4. M - low, right
5. Repeat step 3 and step 4



L=0 #index of array

H=9#index of array

M=(L+H)//2

M= (0+9)//2=4.5 = 4 = 16

16<23 = M low - right = L= M+1 = 4+1=5

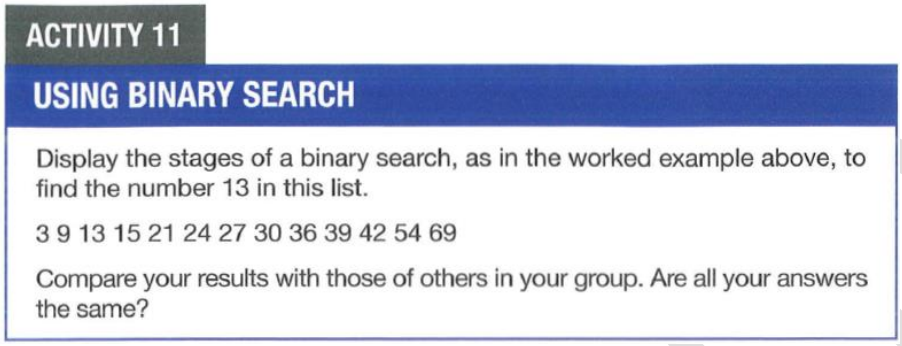
L=5, H=9, M= (5+9)//2=7 = 56

56>23 = M high - left = H = M-1=7-1=6

L=5, H=6, M= (5+6)//2=5.5=5 = 23

23=23, search item found

Activity 11



ACTIVITY 11

USING BINARY SEARCH

Display the stages of a binary search, as in the worked example above, to find the number 13 in this list.

3 9 13 15 21 24 27 30 36 39 42 54 69

Compare your results with those of others in your group. Are all your answers the same?

ACTIVITY 11

3 9 13 15 21 24 27 30 36 39 42 54 69

3 9 13 15 21 24

3 9 13

13

3 9 13 15 21 24 27 30 36 39 42 54 69

13/2=6 #Index of 6 = 27

27>13 = left = 3 9 13 15 21 24

6/2=3 #index of 3 =15

15>13 = left = 3 9 13

3/2=1.5=1 #index of 1 =9

9<13 = right = 13

1/2=0.5=0 #index of 0 = 13

13 = 13 ➔ search item found

WORKED EXAMPLE

If you wanted to find a particular item in a list of 1000 items, these are the best- and worst-case scenarios for the linear search and binary search algorithms.

Linear search A linear search starts at the first item and then works through sequentially. The best case would be if the item is first in the list. The worst case would be if it is last in the list. Therefore, in this example the average would be 500 comparisons.

Binary search The best case would be if the item is in the median position in the list. The search would require only one comparison. For the worst case it would have to choose the following medians until it finally hit the target. (This assumes that the target is always smaller than the median.)

SUMMARY

■There are many algorithms for sorting and searching data.

■The choice of algorithm depends on the data that is to be processed.

■If only a small amount of data needs to be processed, then a simpler, but less efficient search algorithm may be the best choice. The time difference of the search or sort time will be negligible.

CHAPTER 4 - DECOMPOSITION AND ABSTRACTION

computational thinking the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by a computer

- **decomposition** breaking a problem down into smaller, more manageable parts, which are then easier to solve
- **abstraction** the process of removing or hiding unnecessary detail so that only the important points remain

AN EXAMPLE - NOUGHTS AND CROSSES

Input	Output
1. Start the game. 2. Entries for the user. 3. Select a new game or finish	1. **A message to inform the user** when it is their turn. 2. **A message to inform the user** if they try to select a square that has already been used. 3. **A message to inform the user** if the game is a draw. 4. **A message to inform the user** if they or the computer has won. 5. **A message to ask the user** if they want to restart the game or finish.

Activity 12

In a game, each player spins a wheel that is divided into four colours: red, blue, green and yellow.

Each player has to answer a question on a particular topic depending on the colour next to a pointer when the wheel stops.

Red is for science, blue for history, green for general knowledge and yellow for geography.

A player scores two points if they answer correctly on the first attempt and one point for being correct on the second attempt.

The first player to reach 30 points is the winner.

Your task is to design a computer version of the game for up to four players.

You must analyse the problem and list all of the requirements; decompose the problem, list all the sub-problems and write a brief description of each; list all of the input, output and processing requirements.

One of the requirements that will have to be modelled is the spinning of the wheel. Using a written description and pseudocode shows how this could be done.

Input	Output
1. when to start a new game 2. number and names of players 3. when to spin the wheel 4. a player's selected answer 5. whether players want to play again.	**1. A message to inform a player when it is their turn.** **2. A message to inform the player of the outcome of spinning the wheel (question category).** **3. A question plus four possible answers.** **4. A message to inform the player whether their answer is correct or incorrect.** **5. A message to inform the player that they can have another go.** **6. A message to inform the player how many points they have scored.** **7. A message at the end of each round to inform each player of their total score.** **8. A 'game over' message.** **9. A message at the end of the game informing players who has won.** **10. A message to ask whether the players want to play another game or want to finish.**

Processing requirements:

1. Set up question banks for each colour/subject.
2. Flag each question as unused.
3. Establish how many players there are (up to a maximum of four).
4. Set each player's score to 0.
5. Repeat until one player reaches a score of at least 30 or there are no more unanswered questions.
6. Prompt next player to select a subject and simulate spinning the wheel. Display colour and subject selected.
7. Randomly select a question from the remaining unanswered questions in the subject question bank.
8. Display the selected question and four possible answers. Prompt player to select an answer.
9. Receive player's answer. If correct, increment score by 2. If incorrect, prompt player to have a second go. If second attempt successful, increment score by 1.

Subprograms:

- select_category
- display_Q&A
- check_response

- update_score
- mark_asked_questions
- establish_winner
-

One of the requirements that will have to be modelled is the spinning of the wheel. Using a written description **and** pseudocode shows how this could be done.

Algorithm to simulate spinning the wheel to select a colour:

- Select a random number between 0 and 3.
- If the number is 0 then display a red square on the screen.
- If the number is 1 then display a blue square on the screen.
- If the number is 2 then display a green square on the screen.
- Otherwise display a yellow square on the screen.

Pseudocode

```
SET number To RANDOM(3)
IF number = 0 THEN
      colour=red
ELSE
      IF number=1 THEN
             colour =blue
      ELSE
            IF number=2 THEN
                   colour =green
            ELSE
                   colour =yellow
            END IF
      END IF
END IF
SEND “The selected colour is: ” & colour TO DISPLAY
```

SUMMARY

■Computational thinking is an approach to solving problems, such as traffic flow in a city, or how many products a business needs to make and sell to produce a profit. It includes techniques such as decomposition and abstraction.

■Problems are easier to solve if they are decomposed into smaller sub-problems.

■Abstraction is used to remove unnecessary detail to make a problem easier to understand and solve. For example, when modelling traffic flow in a city, unnecessary details could include the colours of the vehicles or the ages of the drivers.

■When designing a solution to a problem the inputs, outputs and processing requirements should be identified at the outset.