2024

COMPUTER SCIENCE

DR. SAW MYAT SANDAR

Pearson Edexcel International GCSEs (O Level)

CHAPTER 1 - SORTING AND SEARCHING ALGORITHMS

Algorithm

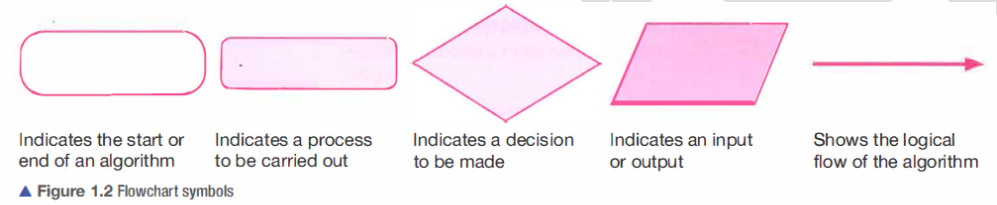
- Step by step instructions/procedures
- Unambiguous
- More than one way

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.

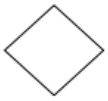
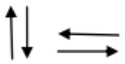
*** ***Algorithms and programs are closely related, but they are not the same.***

Flowchart



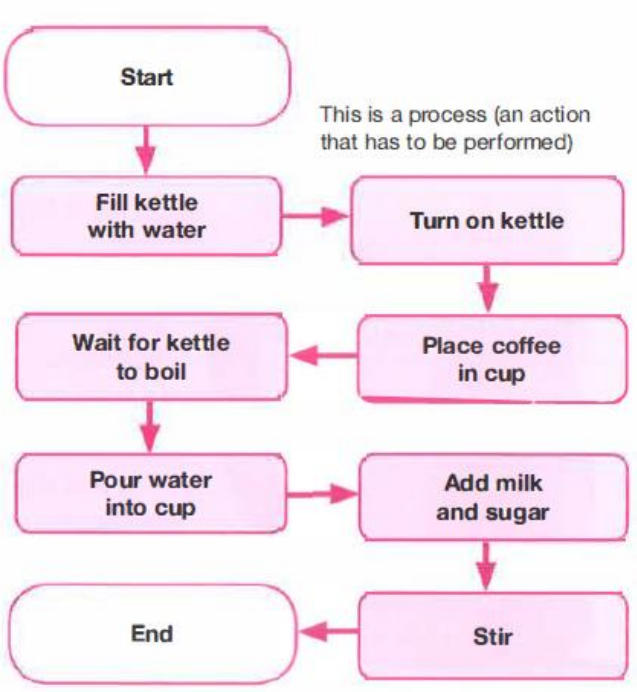
▲ Figure 1.2 Flowchart symbols

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.



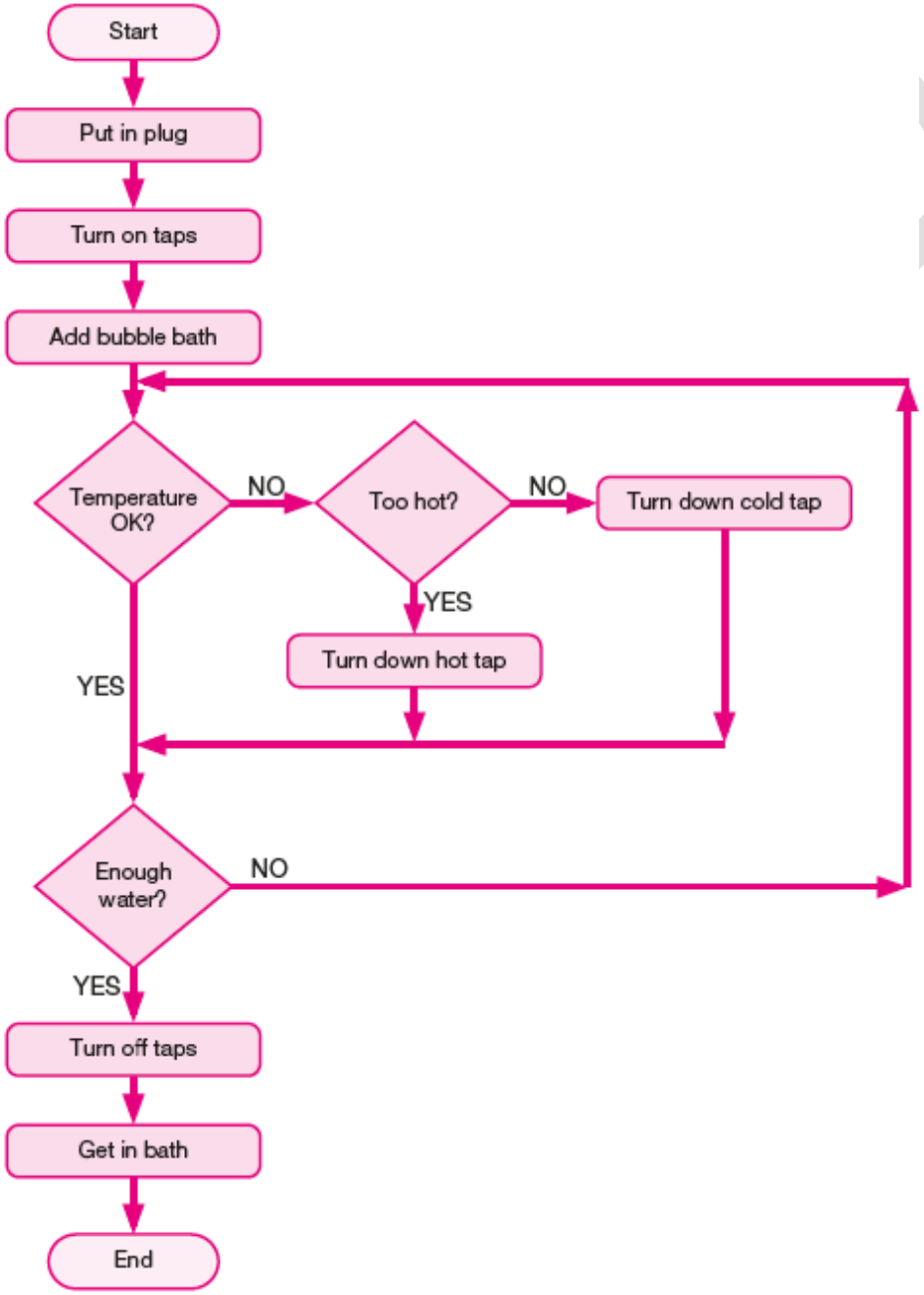
Activity 3

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

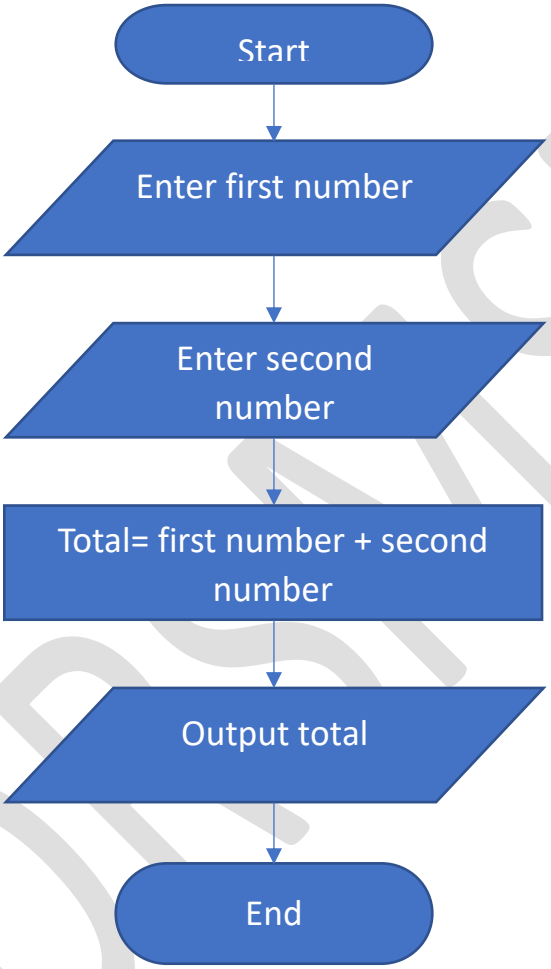
Enter First number.

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.

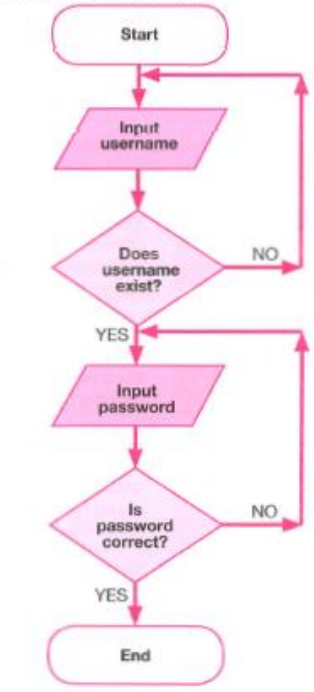
Express this algorithm in pseudocode.

```
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.)

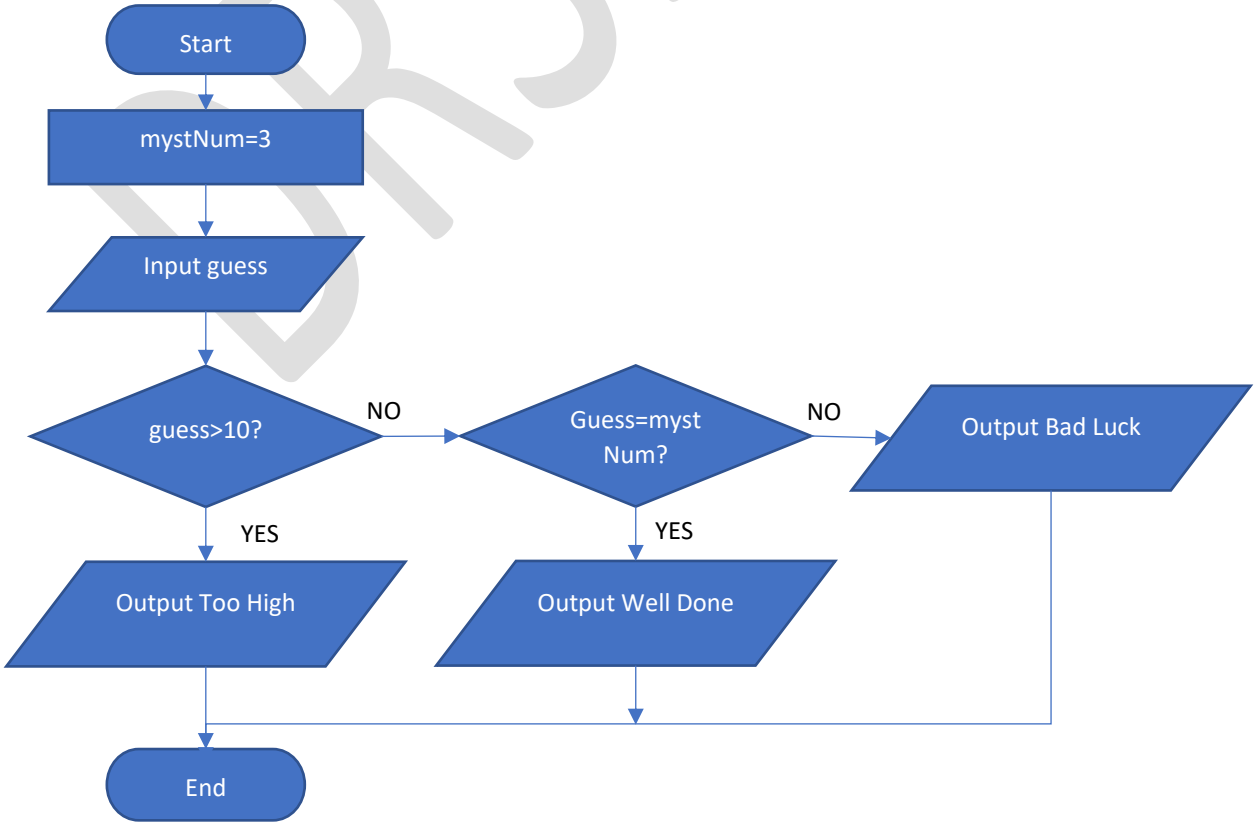
Can you make an algorithm to solve this problem and express it as a written description and a flowchart?

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?

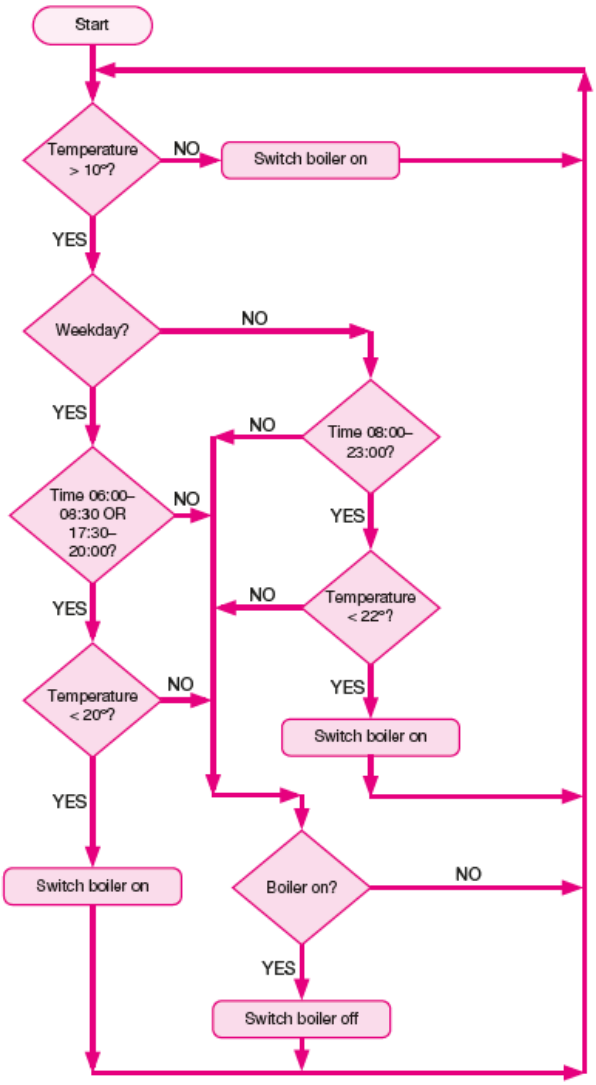
Score	Output
91	**A**
56	**D**
78	**B**
23	D
66	C
0	FAIL

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

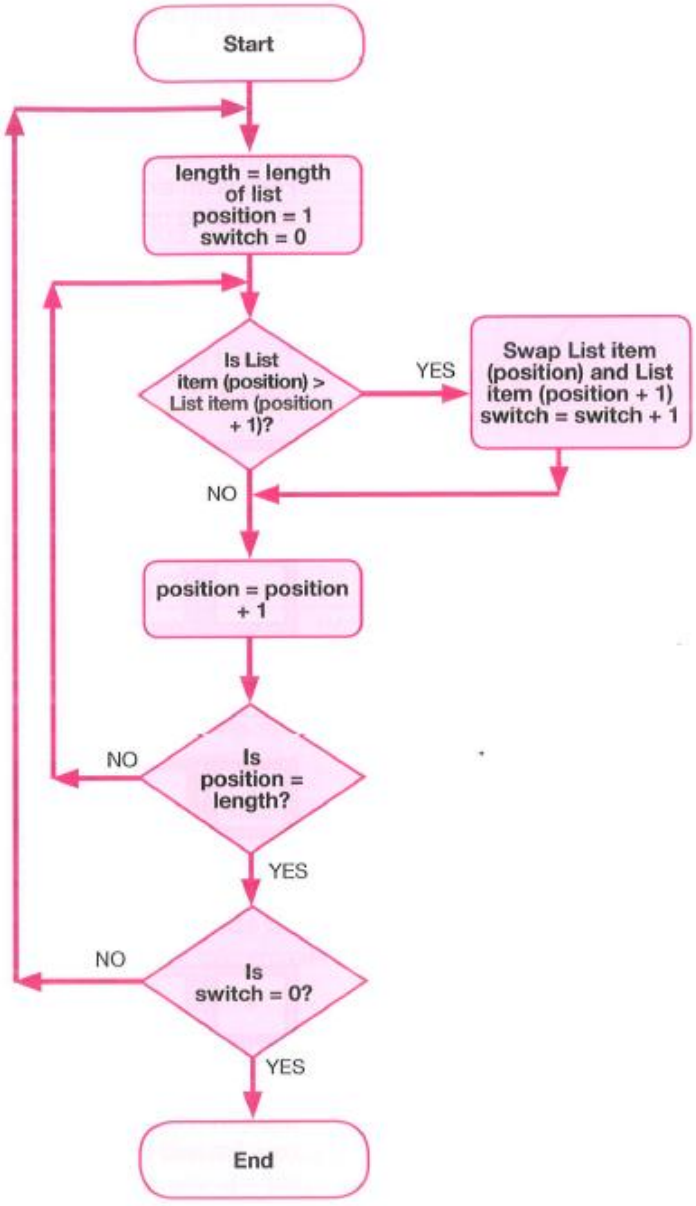
1. Start at the begin of the list
2. Compare the value in position 1 and position 2, if not ascending, swap
3. Compare the value in position 2 and position 3, swap if necessary
4. Continue this process at the end of the list.
5. If swap, repeated step 2 to 4

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

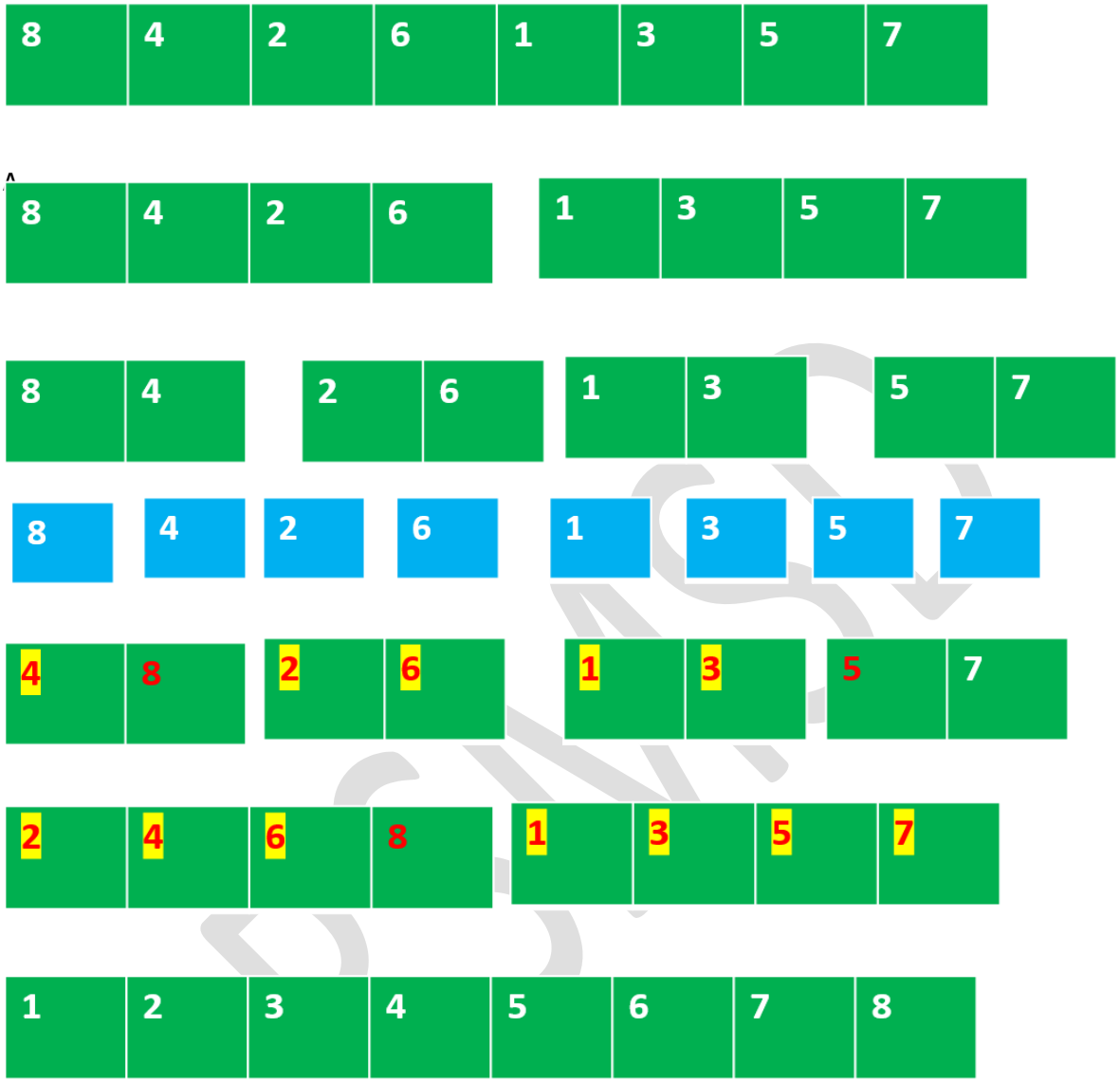
1 2 3 4 6 - ns

1 2 3 4 6 – ns

There is no swap, it is sorted.

MERGE SORT

WORKED EXAMPLE



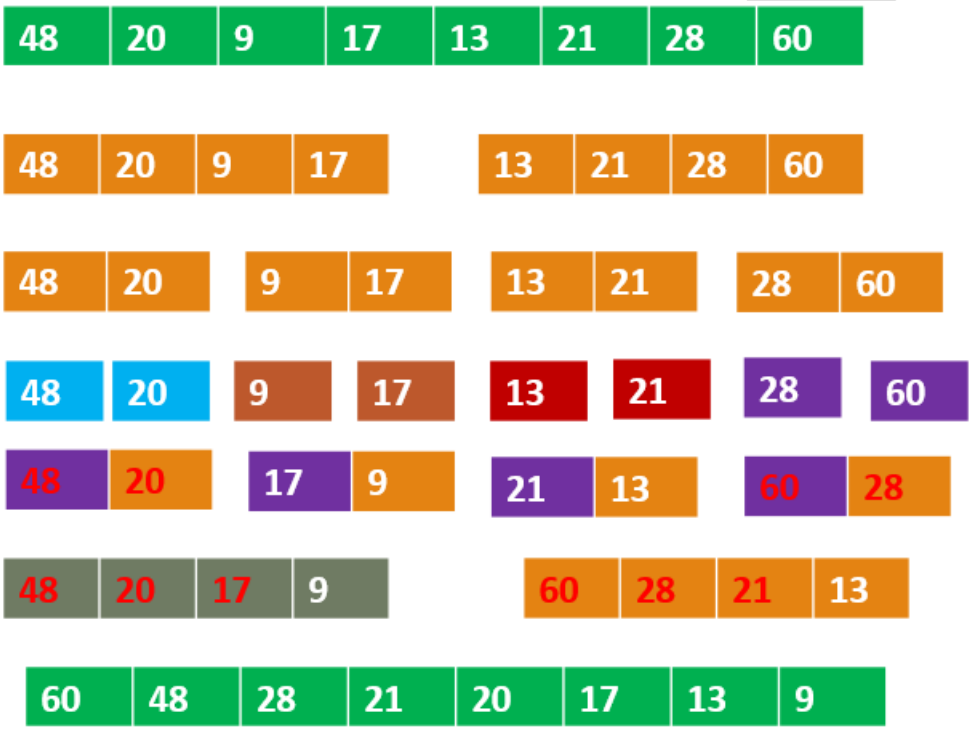
Activity 10

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.

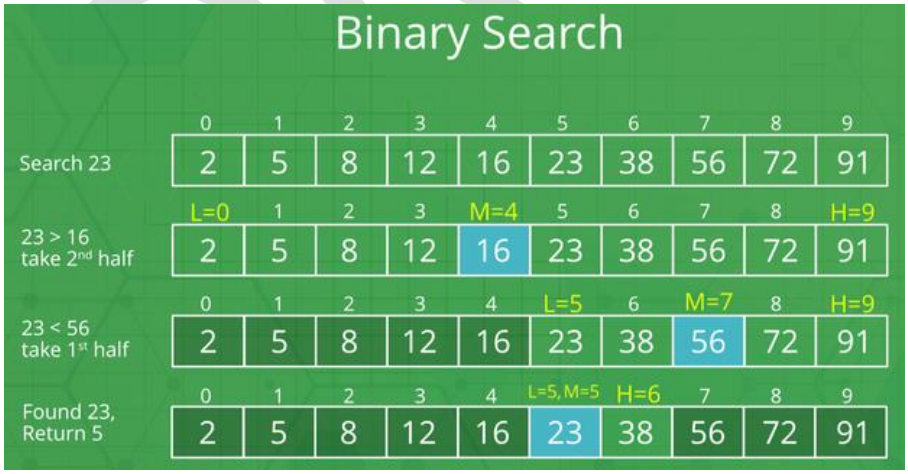
BINARY SEARCH

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

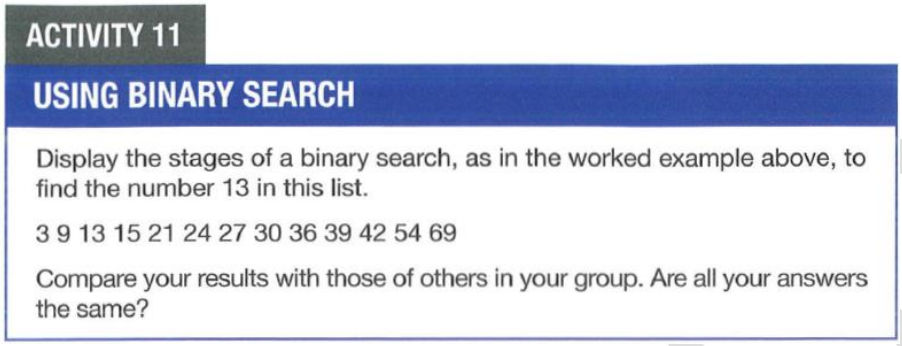
L=5, H=9, M= (5+9)//2=7 = 56

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

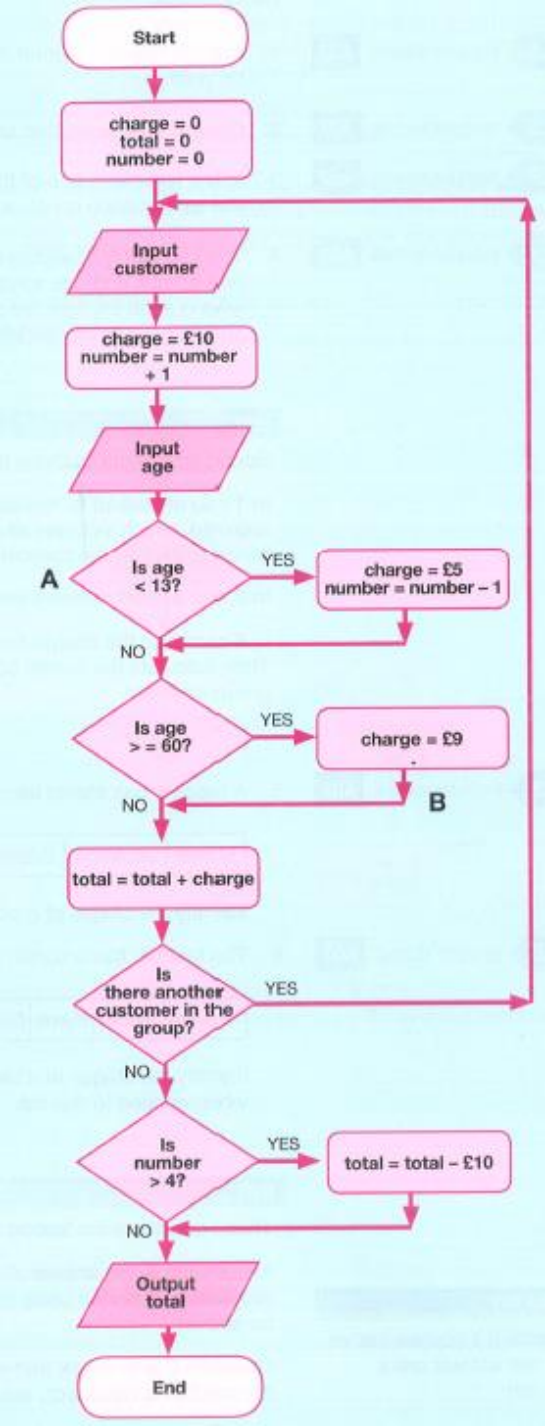
■Computational thinking is an approach to solving problems, such as traffic flow in a city, or how many products a business needs to make and sell to produce a profit. It includes techniques such as decomposition and abstraction.

■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.

UNIT QUESTIONS ***



1. Explain how the algorithm calculates the total amount that should be paid.

2. Give two variables that are used in the algorithm.

3. In the flowchart, two of the constructs are labelled A and B. State the type of each construct.

4. The Lim family is visiting the park. The family consists of two children, one aged 8 and one aged 10, their two parents and their grandfather, who is aged 65. Use the algorithm to calculate how much the family should have to pay for entry.

1.The regular charge is £10,

but children under 13 pay £5

and people aged 60 or above pay £9.

A group consisting of five or more adults gets a discount of £10.

2. The variables used in the flowchart are: charge, total, number, age.

3. Label A denotes a decision to be made and B denotes a process to be carried out.

A=decision to be made

B=process to be carried out

4. (2 × £5) + (2 × £10) + (1x£9) = £39

7 Create an algorithm to calculate the cost of sending a parcel.

If the weight of the parcel is 2 kg or under then the standard charge is $3.

There is then a charge of $2 for each extra kilogram up to 10 kg.

After 10 kg the charge per extra kilogram is $3.

a Display your algorithm as a flowchart.

b Construct your algorithm as pseudocode.

w<=2 kg, cost = $3

w <=10 kg ➔

w = 10 kg

= 2 kg + 8 kg

W – 2

w = 7 kg

= 2 kg + 5 kg

W - 2

Cost = 3 + (w-2)*2

w>10 kg ➔ extra = $2

10 - 2

W - 10

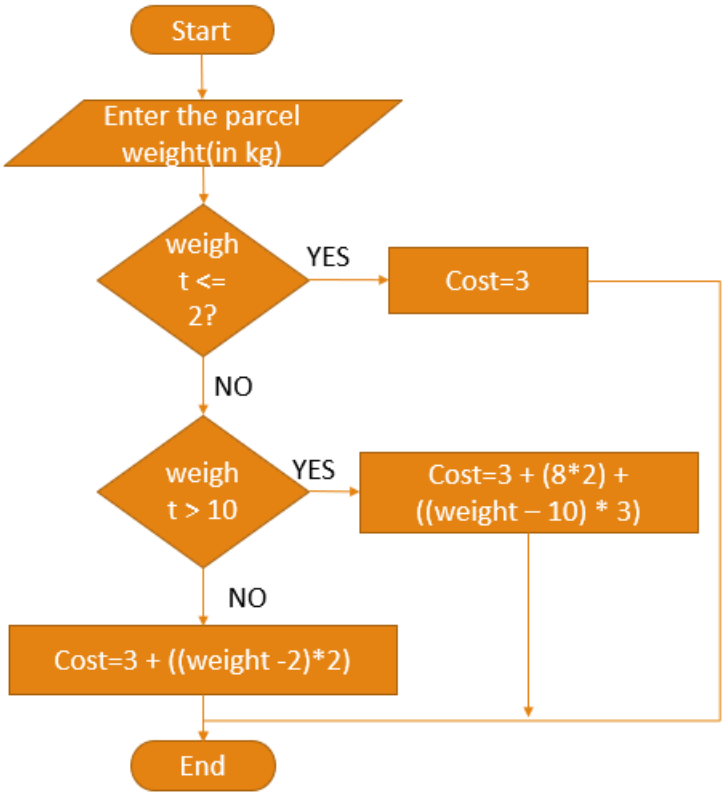
w = 15 kg = 2 kg + 8 kg + 5 kg

w = 12 kg = 2 kg + 8 kg + 2 kg

W - 10

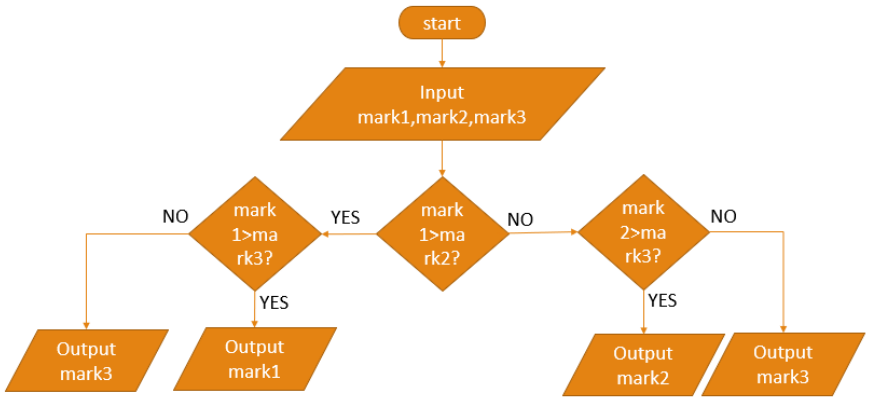
cost = 3 + (8*2) + (w-10)*3

```
SEND ‘ Enter the parcel weight(in kg)’ TO DISPLAY
RECEIVE weight FROM (INTEGER) KEYBOARD
IF weight <= 2 THEN
      SET cost TO 3
ELSE
      IF weight > 10 THEN
            SET cost TO 3 + (8*2) + ((weight – 10) * 3)
      ELSE
            SET cost TO 3 + ((weight -2)*2)
      END IF
END IF
        SEND cost TO DISPLAY
```



8 A learner hands in three homework assignments, which were each given a mark out of 10. All of the marks were different. The following is part of an algorithm to find the highest mark but some of the decision symbols are empty.

Complete the decision symbols and add 'YES' and 'NO' labels where required.



9 A list is made up of the numbers 4, 1, 2, 6, 3, 5.
Identify the steps involved when sorting this list using a **bubble sort algorithm.**

Pass 1

4 1 2 6 3 5 – s

1 4 2 6 3 5 – s

1 2 4 6 3 5 – ns

1 2 4 6 3 5 – s

1 2 4 3 6 5 – s

1 2 4 3 5 6

Pass2

1 2 4 3 5 6 – ns

1 2 4 3 5 6 – ns

1 2 4 3 5 6 – s

1 2 3 4 5 6 - ns

1 2 3 4 5 6 - ns

Pass 3

1 2 3 4 5 6 – ns

1 2 3 4 5 6 – ns

1 2 3 4 5 6 – ns

1 2 3 4 5 6 – ns

1 2 3 4 5 6 - ns

No swap, sorted list

UNIT 3 – DATA

Chapter 12 - BINARY

CONVERT FROM BINARY TO DECIMAL

PLACE VALUES	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	1	0	1	0	1	1	0	1

10101101

=(1x 128)+(0x64)+(1x32)+(0x16)+(1x8)+(1x4)+(2x0)+(1x1)

=128+32+8+4+1

=173

1. Binary to Denary
2. Denary to Binary
3. Addition
4. Two's complement
5. Logical Shift - Multiplication – shift left
6. Logical Shift - Division – shift right
7. Sign and magnitude – 8 bits = x xxx xxxx
8. Arithmetic Shift – Left shift – MSB remain
9. Arithmetic Shift – Right Shift – MSB copy
10. Hexadecimal – to Binary
11. Binary to Hexadecimal

Activity 1

CONVERT FROM BINARY TO DECIMAL

00111001

=(128x0)+(64x0)+(32x1)+(16x1)+(8x1)+(4x0)+(2x0)+(1x1)

=32+16+8+1=57

11000110

=(128x1)+(64x1)+(32x0)+(16x0)+(8x0)+(4x1)+(2x1)+(1x0)

=128+64+0+0+0+4+2+0

=198

10101010

=(128x1)+(64x0)+(32x1)+(16x0)+(8x1)+(4x0)+(2x1)+(1x0)

=128+0+32+0+8+0+2+0

=170

CONVERTING DENARY TO BINARY

Place Values	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
213	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
213	1	1	0	1	0	1	0	1

ACTIVITY 2

What are the binary equivalents of the following denary numbers?

a 69

b 193

c 239

a 69 = 01000101

b 193 = 11000001

a 239 = 11101111

Binary Addition

We are adding 8-bit numbers, and this has caused a problem. All 8 bits have been used and the 1 that was carried over in the last column has nowhere to go - it has been 'carried out'. Therefore, the result of the calculation would be wrong. This is called an **overflow error**.

overflow

error this condition occurs when a calculation produces a result that is greater than the computer can deal with or store. When this happens, the microprocessor is informed that an error has occurred

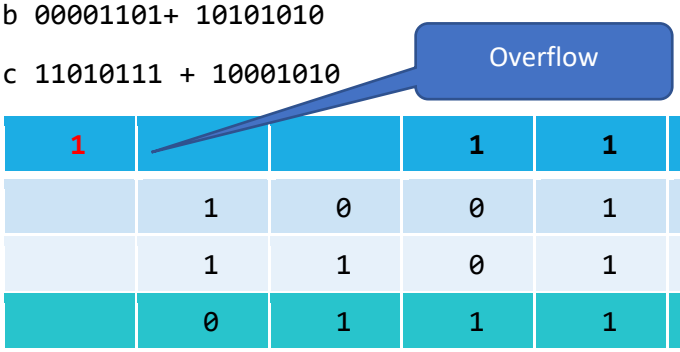
Binary Addition

Carry out the following binary additions.

a 10011010 + 11010111

b 00001101+ 10101010

c 11010111 + 10001010



1			1	1	1	1		
	1	0	0	1	1	0	1	0
	1	1	0	1	0	1	1	1
	0	1	1	1	0	0	0	1

a 10011010 + 11010111 = 101110001

b 00001101 + 10101010 = 10110111

c 11010111 + 10001010 = 101100001

TWO'S COMPLEMENT (binary negative)

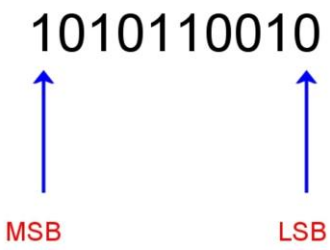
In a multiple-bit binary number, the left-most bit (the one with the greatest value) is called the most significant bit (MSB). We can use this to represent signed integers.

It is set to 1 for negative and 0 for positive.

SIGN AND MAGNITUDE

MSB ➔ 0 ➔ positive ➔ 00011100 ➔ +28

MSB ➔ 1 ➔ negative ➔ 10011100 ➔ -28



The most significant bit is being used for the sign, so the largest positive number represented by a byte is 127.

To find the two's complement of a binary number:

■flip all of the bits - change 1 s to 0s and 0s to 1 s

■add 1 to the result.

28	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
Original Data of 28	0	0	0	1	1	1	0	0
Flip	1	1	1	0	0	0	1	1
Add 1								1
Two's complement -28	1	1	1	0	0	1	0	0

Method 2 – Place Values

-28 ➔ -128 + 100

-128 + 64 + 32 + 4

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
-128	64	32	16	8	4	2	1
1	1	1	0	0	1	0	0

Worked Example - Adding 28+(-5)

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
28	0	0	0	1	1	1	0	0

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
5	0	0	0	0	0	1	0	1
Flip	1	1	1	1	1	0	1	0
Add 1								1
-5	1	1	1	1	1	0	1	1

	1	1	1	1	1				
28		0	0	0	1	1	1	0	0
-5		1	1	1	1	1	0	1	1
23	discard	0	0	0	1	0	1	1	1

Activity 4

ACTIVITY 4

TWO'S COMPLEMENT REPRESENTATION

1 What is the two's complement representation of the following denary numbers?

a –113
b –56
c –90

2 Showing your working, carry out the binary addition of 90 + (–33).

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
113	**0**	**1**	**1**	**1**	**0**	**0**	**0**	**1**
flip	1	0	0	0	1	1	1	0
Add 1								1
-113	1	0	0	0	1	1	1	1

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
56	**0**	**0**	**1**	**1**	**1**	**0**	**0**	**0**
flip	1	1	0	0	0	1	1	1
Add 1								1
-56	1	1	0	0	1	0	0	0

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
90	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**
flip	1	0	1	0	0	1	0	1
Add 1								1
-90	1	0	1	0	0	1	1	0

2 90 + (–33) = 01011010 + 11011111 = 00111001

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
33	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**
flip	1	1	0	1	1	1	1	0
Add 1								1
-33	1	1	0	1	1	1	1	1

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
90	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**
-33	1	1	0	1	1	1	1	1
	1		1	1	3	2	2	1
57	0	0	1	1	1	0	0	1

BINARY SHIFTS/LOGICAL SHIFT

In binary left and right, binary shifts can be used for multiplication and division by powers of 2.

Left Shift

The left-most bits drop off the end and are replaced by 0s at the right.

00010100 - left shift – 4 - 2^2
Left shift – multiply

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
	0	0	0	1	0	1	0	0
	0	1	0	1	0	0	x	x
	0	1	0	1	0	0	0	0

Check

00010100 = 20

2^2=4

20*4=80

01010000

Right Shift

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
20	0	0	0	1	0	1	0	0
2 right shift	x	x	0	0	0	1	0	1
	0	0	0	0	0	1	0	1

Check

00010100 = 20

2^2=4

20/4=5 ➔ 00000101 = 5

Activity 5

ACTIVITY 5

LOGICAL SHIFTS ON UNSIGNED INTEGERS

What are the results of the following shifts on unsigned integers?

a 00111010 * 2^3 b 10011101 / 2^4

(a) 00111010 (58) * 2^3 (8)

Multiply – Left shift

	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	256	128	64	32	16	8	4	2	1
00111010=58		0	0	1	1	1	0	1	0
111010000=464	1	1	1	0	1	0	x	x	x
	1	1	1	0	1	0	0	0	0

Check

58*8= 464 - 8 bits – 0 ~ 7 ➔ Min = 0000 0000 = 0 to Max = 1111 1111= 255

(b) 10011101 / 2^4 = 00001001

Divide – Right shift

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
10011101 = 157	1	0	0	1	1	1	0	1
	x	x	x	x	1	0	0	1
00001001=9	0	0	0	0	1	0	0	1

Check

10011101 / 2^4 = 00001001

157/16=9

Actually ➔ 157/16= 9.8125

Right Shift ➔ 157/16 = 9

*** A logical shift on integers can make a number less precise. For example, if we divide 00100001 (33 in denary) by 2 the digits would be shifted one place to the right.

ARITHMETIC SHIFTS

Arithmetic shifts are used with signed numbers expressed in two's complement format

LEFT ARITHMETIC SHIFT

A left arithmetic shift is identical to a left logical shift except that the left-most bit **(MSB) is not included** because it must remain in place to indicate the sign.

Arithmetic Shift - Calculate the product of -36 * 2

1	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
36	0	0	1	0	0	1	0	0
Flip	1	1	0	1	1	0	1	1
Add 1								1
-36	1	1	0	1	1	1	0	0

11011100 = -128+64+16+8+4= -36

-36*2	1	1	0	1	1	1	0	0
-72	1	0	1	1	1	0	0	0

10111000 = -128+32+16+8= -72

A left arithmetic shift is identical to a left logical shift except that the left-most bit (MSB) is not included

RIGHT ARITHMETIC SHIFT

When dividing signed binary numbers in two's complement format by powers of 2, the bits are shifted to the right.

They are replaced at the left by copies of the MSB.

Arithmetic Shift +➔ -72/2^2 (-72/4)

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1
72	0	1	0	0	1	0	0	0
Flip	1	0	1	1	0	1	1	1
Add 1								1
-72	1	0	1	1	1	0	0	0
2 right shift	x	x	1	0	1	1	1	0
	1	1	1	0	1	1	1	0

MSB

replaced at the left by copies of the MSB

11101110= **- 128**+64+32+8+4+2= -18 (-72/4= -18)

Activity 6

ACTIVITY 6

ARITHMETIC SHIFTS

1 Show the following division:
   - a 10010000 / 2^2 in binary
   - b 11110110 / 2^1 in binary
   - c 11000000 / 2^3 in binary.

2 Show that your answers are the expected results.

a 10010000 / 2^2 = 11100100 (−112 / 4 = −28)

b 11110110 / 2^1 = 11111011 (−10 / 2 = −5)

c 11000000 / 2^3 = 11111000 (−64 / 8 = −8)

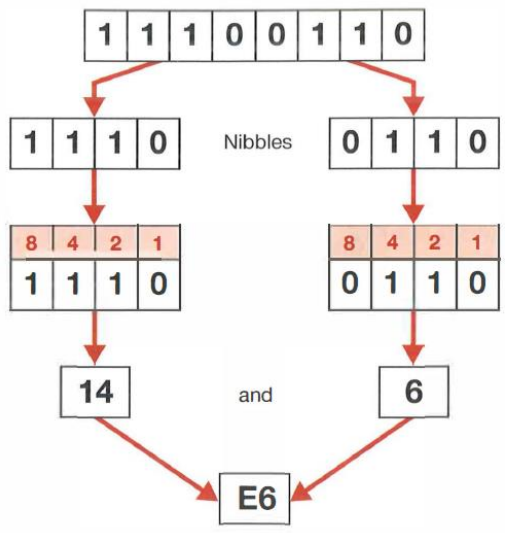
(a)	(b)	(c)
10010000 / 2^2 10010000 11100100 -128+64+32+4=-28 **Check** 10010000 = -112 2^2 =4 -112/4=-28	11110110 / 2^1 11110110 11111011 -128+64+32+16+8+2+1=-5 **check** 11110110 = -10 2^1 =2 -10/2=-5	11000000 / 2^3 11000000 ***111***11000 -128+64+32+16+8+4=-8 **check** 11000000 = - 64 2^3 =8 -64/8 =64

HEXADECIMAL NUMBERS

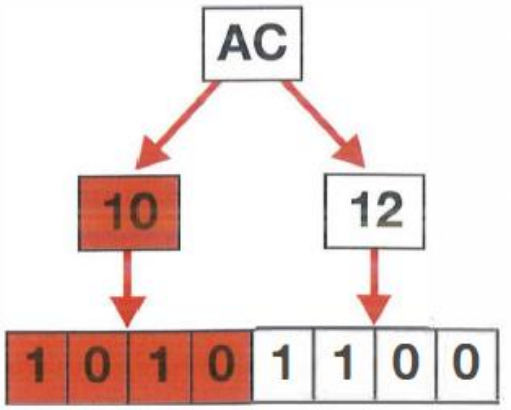
Binary				Decimal	Hexadecimal
8	4	2	1		
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	8
1	0	0	1	9	9
1	0	1	0	10	A
1	0	1	1	11	B
1	1	0	0	12	C
1	1	0	1	13	D
1	1	1	0	14	E
1	1	1	1	15	F

*** hexadecimal numbers is to describe colours used in computer graphics

*** #FF0000 = 11111111000000000000000.



▲ Figure 3.1 Converting binary numbers to hexadecimal



▲ Figure 3.2 Converting hexadecimals

Activity 7

ACTIVITY 7

HEXADECIMAL AND BINARY CONVERSION

1 How can the following binary numbers be expressed in hexadecimal?

a 01101110
b 10011100
c 00101010

2 How can the following hexadecimal numbers be expressed in binary?

a A6
b 9C
c 2D

ACTIVITY 7

1 a 01101110 = 6E

b 10011100 = 9C

c 00101010 = 2A

2 a A6 = 10100110

b 9C = 10011100

c 2D = 00101101

CHECKPOINT ***

CHAPTER 13 DATA REPRESENTATION

The defined list of characters recognised by a computer's hardware and software is known as its character set.

an industry standard is needed

This standard is provided by the American Standard Code for Information Interchange, or ASCII code.

ASCII code consisted of **7 bits** and so **128 characters**

Important groups are

1. 65 to 90 for the upper-case alphabetic characters and
2. 97 to 122 for their lower-case equivalents
3. The numeric characters 0 to 9 are represented by codes 48 to 57.

D	B	C	D	B	C	D	B	C	D	B	C
32	00100000	space	57	00111001	9	82	01010010	R	107	01101011	k
33	00100001	!	58	00111010	:	83	01010011	S	108	01101100	l
34	00100010	"	59	00111011	;	84	01010100	T	109	01101101	m
35	00100011	#	60	00111100	<	85	01010101	U	110	01101110	n
36	00100100	$	61	00111101	=	86	01010110	V	111	01101111	o
37	00100101	%	62	00111110	>	87	01010111	W	112	01110000	p
38	00100110	&	63	00111111	?	88	01011000	X	113	01110001	q
39	00100111	'	64	01000000	@	89	01011001	Y	114	01110010	r
40	00101000	(	65	01000001	A	90	01011010	Z	115	01110011	s
41	00101001	)	66	01000010	B	91	01011011	[	116	01110100	t
42	00101010	*	67	01000011	C	92	01011100	\	117	01110101	u
43	00101011	+	68	01000100	D	93	01011101	]	118	01110110	v
44	00101100	,	69	01000101	E	94	01011110	^	119	01110111	w
45	00101101	-	70	01000110	F	95	01011111	_	120	01111000	x
46	00101110	.	71	01000111	G	96	01100000	`	121	01111001	y
47	00101111	/	72	01001000	H	97	01100001	a	122	01111010	z
48	00110000	0	73	01001001	I	98	01100010	b	123	01111011	{
49	00110001	1	74	01001010	J	99	01100011	c	124	01111100 |	
50	00110010	2	75	01001011	K	100	01100100	d	125	01111101	}
51	00110011	3	76	01001100	L	101	01100101	e	126	01111110	~
52	00110100	4	77	01001101	M	102	01100110	f	127	01111111	DEL
53	00110101	5	78	01001110	N	103	01100111	g			
54	00110110	6	79	01001111	O	104	01101000	h			
55	00110111	7	80	01010000	P	105	01101001	i			
56	00111000	8	81	01010001	Q	106	01101010	j			

KEY: D = denary, B = binary, C = character

▲ Figure 3.3 The printable characters of the 7-bit ASCII code

Activity 5

ACTIVITY 8

FINDING ACSII CODES FOR CHARACTERS

Using the table in Figure 3.3, write down the ASCII codes for the characters in the following phrase:

ASCII code

The ASCII code for 'ASCII code.' is:

01000001 01010011 01000011 01001001 01001001 00100000

01100011 01101111 01100100 01100101 00101110

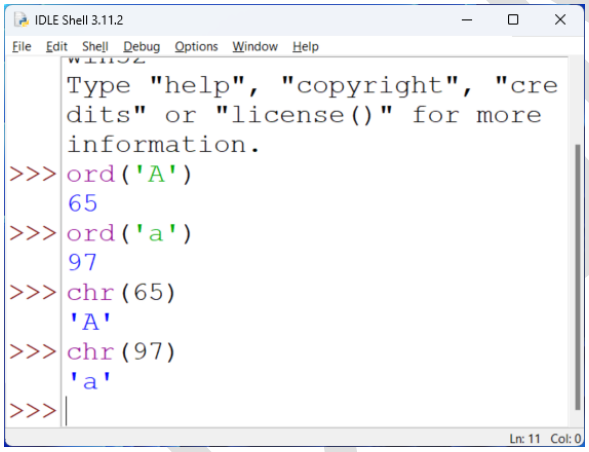
Encrypting

The **encrypted message** is called a **cipher**

***** The Pearson Edexcel pseudocode does not have these functions built in, but in the Python programming language they are ord () and chr ().**

ord ('c') would return 99, and

chr (100) would return 'd'.



```
SEND "Please enter the text" TO DISPLAY
RECEIVE myString FROM (STRING) KEYBOARD
FOR index FROM 0 TO LENGTH(myString) - 1 DO
    SET number TO ord(myString[index])
    SEND number TO DISPLAY
END FOR
```

#The function is called using the character in the string as an argument.

By using the `chr()` function you can build up a string using characters entered by their denary codes.

The following algorithm will allow you to enter ten numbers which will be converted to characters and appended to the string.

```
SET myString TO ""                                  #This will create an empty string.
FOR index FROM 0 TO 10 DO
   SEND "Please enter a number in the range 65 to 90 or in the range 97 to 122"
TO DISPLAY
    RECEIVE number FROM (INTEGER) KEYBOARD
    SET character TO chr(number)
    SET myString TO myString + character
END FOR
```

```
FUNCTION ord(character)
BEGIN FUNCTION
    SET arrayAscii TO [["!"][33], ["""][34], ...["z"][122]]   #A two-dimensional array containing all of the characters with their denary codes.
    FOR index FROM 0 TO LENGTH(arrayAscii) - 1 DO
        IF character = arrayAscii[index, 0] THEN
            code = arrayAscii[index, 1]
        END IF
    END FOR
RETURN code
END FUNCTION
```

ACTIVITY 9

```
FUNCTION chr(number)
BEGIN FUNCTION
      SET arrayNumb2Ascii TO [[32, ''], [33, '!'], [34, '"'], [35, '#'], [36,'$'], [37,
      '%'], [38, '&'], .....[125,'}']]
      FOR index FROM 0 TO LENGTH(arrayNumb2Ascii) - 1 DO
          IF number = arrayNumb2Ascii[index, 0] THEN
                character = arrayNumb2Ascii[index, 1]
          END IF
      END FOR
      RETURN character
END FUNCTION
```

#A 2-dimensional array containing all of the denary ASCII codes and characters

Activity 10

ACTIVITY 10

ENCRYPTION

1. Create an algorithm that will encrypt the following sentence: 'The ASCII code represents characters.'
2. Encrypt it with a shift of 3 to the right (i.e. A should be encrypted as D). (Remember: the spaces and the full stop should not be changed.)
3. Output the encrypted text.
4. Present your algorithm as pseudocode or code and test it in the programming language you are studying.

```
message=input("Enter the message to encrypt:")
shift=int(input("Enter the size of the shift:"))
secreteMessage=""
for character in message:
    number=ord(character)#ord('a')=97
    if character.lower() in 'abcdefghijklmnopqrstuvwxyz':#a - 97 ==
97+3=100 = d
        number+=shift
        if character.isupper():#A
            if number>ord('Z'):
                number-=26#Z - 90 >> 90+3=93 >> ] >> 93-26 = 67 = C
            elif number<ord('A'):
                number+=26
        else:
            if number>ord('z'):#z = 122 >> 122+3=125
                number-=26
            elif number<ord('a'):
                number+=26
        print(chr(number))
        secreteMessage=secreteMessage+chr(number)
        print(secreteMessage)
    else:
        secreteMessage=secreteMessage+character
        print(secreteMessage)
```

UNICODE

In the ASCII code, there are 96 printable characters but there was always a need for more characters in order to accommodate foreign languages, mathematical symbols and special symbols for drawing pictures.

As computers process data in 8-bit bytes, ASCII was extended to an 8-bit code, which allows 256 codes. Unfortunately, there was no standardisation.

To overcome the problem of **multiple versions of ASCII**, the Unicode Consortium was founded to develop and promote a **Unicode standard.** This represents and handles text in most of the world's writing systems using 2 or 4 bytes. (16 bits or 32 bits)

For compatibility and because ASCII was the recognised standard, Unicode characters 0 to 127 are the same as ASCII. In English-language documents, Unicode is represented using the same codes as in 8-bit ASCII.

REPRESENTATION OF BITMAP IMAGES

*** pixel short for 'picture element' – ppi/ppc

*** smallest single point of colour in a graphic image resolution

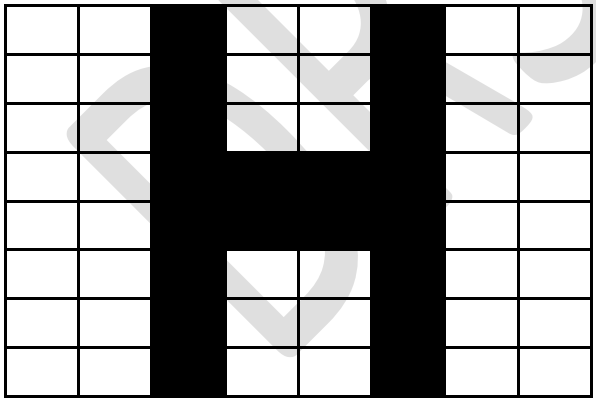
*** basic unit of a bitmap image is the pixel

IMAGE SIZE AND RESOLUTION

*** ***The greater the number of pixels within a given area the higher the resolution and the more detail shown.***

*** *The lower the resolution, the less the amount of detail seen.*

If 0 represents black and 1 white, the letter 'H' could be encoded with the following bit pattern.



```
11011011
11011011
11011011
11000011
11000011
11011011
11011011
11011011
```

FILE SIZE FOR A BITMAP IMAGE

Width x Height x Colour depth

Activity 11

Image File Size

Create expressions and calculate the file sizes of the following images. Express the sizes in bits and bytes.

a A 256-colour image with a size of 640 x 480 pixels.

b A true-colour image with a size of 640 x 480 pixels.

(a) 256 colour = 2^8 = 8 bits

640x480x8=2 457 600 bits = 2457600/8 =307200 bytes (640x480x8)/8 bytes

(b) true-colour = 24 bits

640x480x24= 7 372 800 bits= 7 372 800 /8 = 921 600 bytes

FIDELITY

Fidelity – higher the sample rate – higher the fidelity

CD – 44 100 (44.1 kHz)

Blu ray – 96 000 (96kHz)

DIGITAL AUDIO FILE SIZES

The size of a digital audio sound file depends on the following:

- sample rate per second
- bit depth
- duration of recording
- number of channels - mono (one channel} or stereo (two channels}.

file size in bits = sample rate * bit depth * duration (in seconds) * number of channels

CALCULATING FILE SIZE FOR AUDIO - WORKED EXAMPLE

Number of samples per second = 44 100

Bit depth = 16 bits

Duration = 2.5 minutes

Number of channels = 2

The size of the file is found from the following formula:

file size in bits = sample rate * bit depth * duration (in seconds) * number of channels

Therefore, the file size of the above recording is:

44 100 * 16 * 2.5 * 60 * 2

= 211 680 000 bits

= 211 680 000/8 bytes

= 26 460 000 bytes.

Activity 12

ACTIVITY 12

AUDIO FILE SIZES

What is the file size of a stereo recording of three minutes' duration with a sample rate of 44 100 and a bit depth of 24 bits? Give your answer in bits and bytes.

ACTIVITY 12

44 100 × 24 × 3 × 60 × 2 = 381 024 000 bits = 47 628 000 bytes

CHECKPOINT ***

CHAPTER 14 DATA STORAGE AND COMPRESSION

Universal use of digital devices, huge amounts of data are being generated and stored

All data consists of bits - is 1 and 0s

The smallest unit = binary digit ➔ bit

4 bits = nibble ➔ hexadecimal

8 bits = byte ➔ store data

*** calculated the size of an example image file - typical **byte** is equivalent to 8 bits

*** **binary to hexadecimal**, you used a unit called a **'nibble'**, which is equal to **4** bits.

> *** Microsoft WindowsÂ® the image file size would be shown as 34.9 MB, while in UbuntuÂ® it would be shown as 36.6 MB.

DECIMAL PREFIX				BINARY PREFIX			
UNIT	SYMBOL	MAGNITUDE	SIZE	UNIT	SYMBOL	MAGNITUDE	SIZE
kilobyte	KB	10^3 bytes	1000 bytes	kibibyte	KiB	2^{10} bytes	1024 bytes
megabyte	MB	10^6 bytes	1000 kilobytes	mebibyte	MiB	2^{20} bytes	1024 kibibytes
gigabyte	GB	10^9 bytes	1000 megabytes	gibibyte	GiB	2^{30} bytes	1024 mebibytes
terabyte	TB	10^{12} bytes	1000 gigabytes	trebibyte	TiB	2^{40} bytes	1024 gigibytes

▲ Table 3.1 Decimal and binary prefixes

DECIMAL PREFIX	BINARY PREFIX
kilobyte (KB)	kibibyte (KiB)
megabyte (MB)	mebibyte (MiB)
gigabyte (GB)	gibibyte (GiB)
terabyte (TB)	tebibyte (TiB)

▲ **Table 3.2** The International Electrotechnical Commission assigned new names to the binary prefix units

An image file with a size of 36 000 000 bytes would be 36 megabytes or 34.3322 mebibytes.

Mega = 36000000/1000 = 36

Mebi = 36000000/1024 = 34.33

Activity 13

ACTIVITY 13

STORAGE CAPACITY

1 A hard disk is described as having a storage capacity of 1.5 TB. What is this in:
   a gigabytes
   b megabytes
   c kilobytes
   d terabytes?

2 An image file has a size of 363 143 213 bits. Create an expression to convert this size to mebibytes and show the result.

1 a 1.5 TB = 1500 GB b 1.5 TB = 1500 000 MB c 1.5 TB = 1 500 000 000 KB d 1.5 TB = 1.5 TB	a 1.5 TB = 1536 GiB b 1.5 TB = 1 572 864 MiB c 1.5 TB = 1 610 612 736 KiB d 1.5 TB = 1.5 TiB

(given in the question. TB means terabytes)

2 363 143 213 bits

= 363 143 213 /8 bytes

= 45,392,901.625 byes

= 45,392,901.625 B/1024 = 44,329.00549 KB

= 44,329.00549316406 KB/1024 = 43.29 MB

43.2 mebibytes.

DATA COMPRESSION

Compression algorithms are used to make the files as small as possible. There are two types of compression - lossless and lossy compression.

LOSSLESS COMPRESSION

*** lossless compression is used, no data is lost and the original file can be restored

*** especially useful for files

*** there are many words that are used more than once – redundancy

*** Lossless compression is essential for text

RUN-LENGTH ENCODING (RLE)

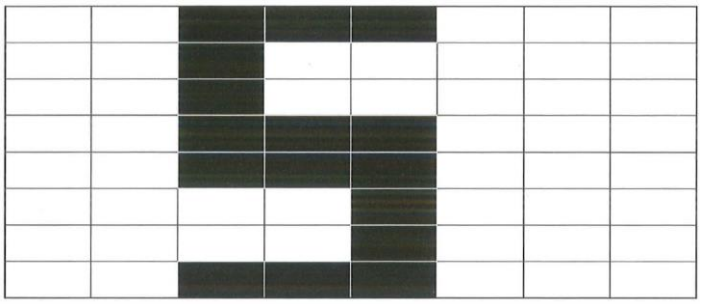
Run-length encoding is used to reduce the size of a repeating string of items

For example, the following string of letters:

cccmmmmmsssssdddcccccc – 22 bytes(B)

would be represented by:

3c5m5s3d6c – 10 bytes(B)



▲ **Figure 3.9** A bitmap diagram of the number 5

Lossless – run length encoding – cccbbb=3c3b – text / program

Lossy – remove – audio/image

CODE	RLE VERSION	SIZE OF CODED VERSION
wwbbbwww	2w3b3w	6
wwbwwwww	2w1b5w	6
wwbwwwww	2w1b5w	6
wwbbbwww	2w3b3w	6
wwbbbwww	2w3b3w	6
wwwwbwww	4w1b3w	6
wwwwbwww	4w1b3w	6
wwbbbwww	2w3b3w	6
64 bytes		**48 bytes**

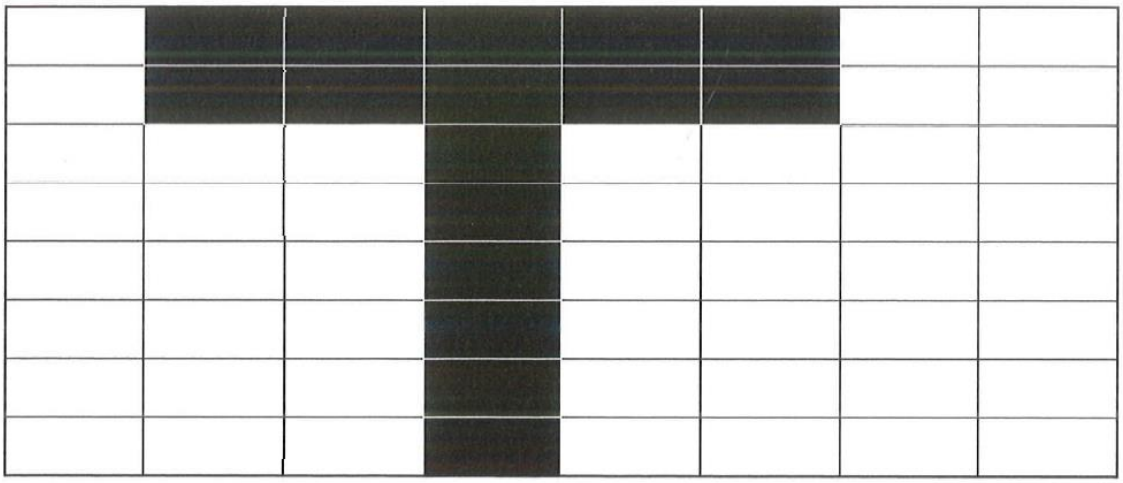
▲ **Table 3.3** Run-length encoding of the image of the number 5 in Figure 3.9

Activity 14

ACTIVITY 14

RUN-LENGTH ENCODING

Show the effect of applying run-length encoding to the graphic in Figure 3.10. Set out your answer as in Table 3.3.



▲ **Figure 3.10** A bitmap graphic of the letter T

ACTIVITY 14

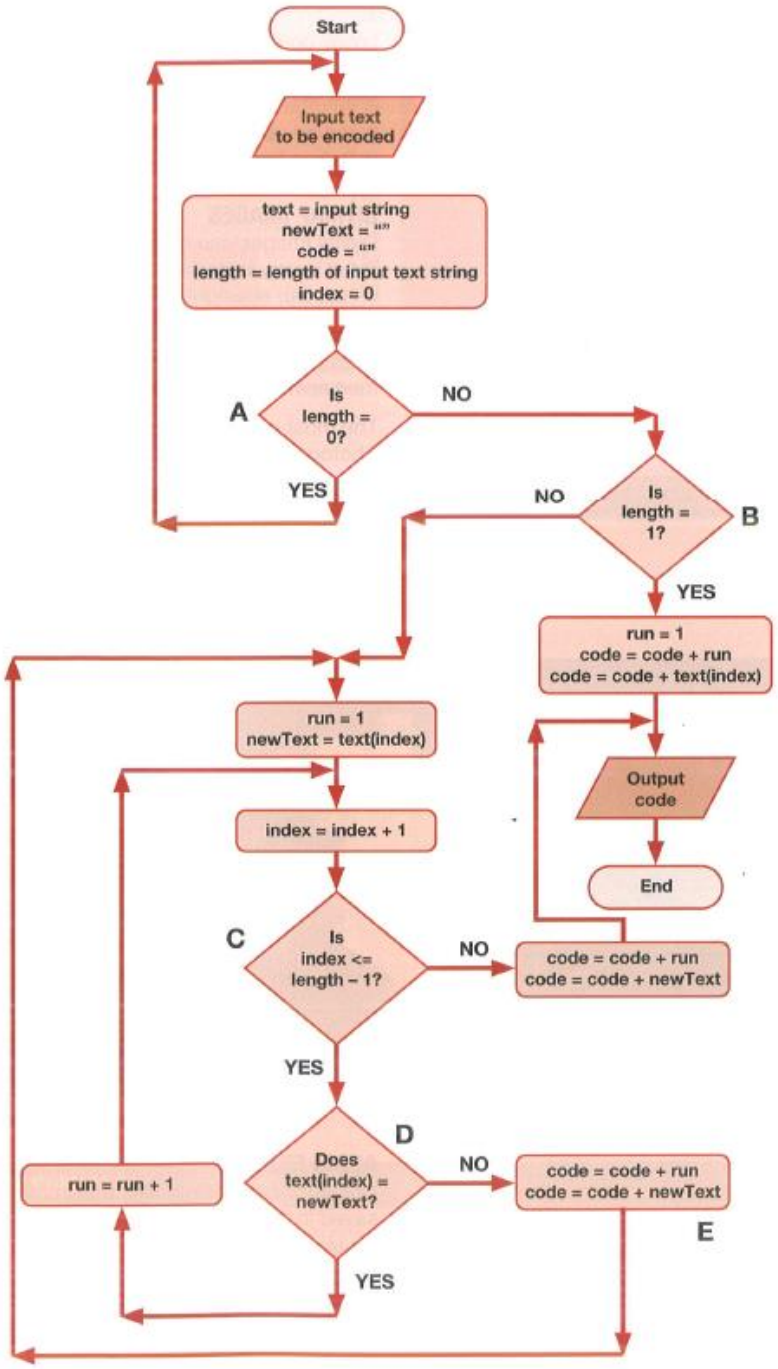
CODE	RLE VERSION	SIZE OF CODED VERSION
wbbbbbww	1w5b2w	6
wbbbbbww	1w5b2w	6
wwwbwwww	3w1b4w	6
wwwbwwww	3w1b4w	6
wwwbwwww	3w1b4w	6
wwwbwwww	3w1b4w	6
wwwbwwww	3w1b4w	6
wwwbwwww	3w1b4w	6
64 bytes		48 bytes

Activity 15

1 Six variables are used in the flowchart in Figure 3.11. List the variables and the data items they represent.

2 Five items are labelled A to E in the flowchart. Match each item below with the statement that describes it (e.g. A3):

1 This checks to see if the end of the input text has been reached.
2 If there is only one character, it and its run length are added to the output file.
3 This adds the character and its run length to the output string.
4 This checks to see that some text has actually been entered.
5 This checks to see if the next character in the input string is the same as or different from the one that is being checked.



1

VARIABLES	DATA ITEMS
text	Input string
newText	The next character in the input string to be evaluted
code	The encoded string
length	Length of input string
run	The number of repetitions of a character
index	Index position of character in input string

2 A=4, B=2, C=1, D=5, E=3

LOSSY COMPRESSION

Lossy compression decreases the file size by deleting some of the data. The original file therefore cannot be re-formed entirely when it is decompressed.

So, it ***cannot*** be used with ***text or program files***.

It can be used for bitmap image and audio files where we often cannot notice that data has been removed.

BITMAP IMAGES

*** tiny differences in colour are wasted

*** eyesight is not capable of distinguishing these small differences.

*** lossy compression algorithm analyses all of the data in the image

*** rewrite the file using fewer bits.

*** most commonly used compression technique - Joint Photographic Experts Group and produces **JPEG/jpg**

AUDIO FILES

*** Uncompressed audio files - Waveform Audio (WAV) format - 3-minute - 30 MB.

*** A 30 MB WAV file can be compressed to a 3 MB MP3

*** Much of the data in an audio file encodes tones and frequencies that our ears cannot hear

CHECKPOINT ***

CHAPTER 15 ENCRYPTION

Plain text – the text that is to be encrypted

Ciphertext - the encrypted plaintext

keystream - the characters that are combined with the plaintext to produce an encrypted message

keyword - the text that is chosen to generate the keystream

ASYMMETRIC ENCRYPTION

encrypts and decrypts data using two different keys

two keys – public key and private key

A message encrypted with a particular public key can only be decrypted by the corresponding private key.

SYMMETRIC ENCRYPTION

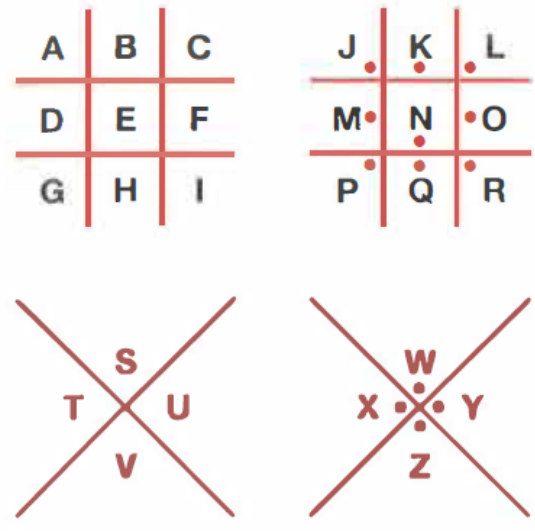
encrypt and decrypt a message using the same key

This is the method used by a HTTPS connection.

4 Types of Cipher Method

1. PIGPEN CIPHER
2. THE CAESAR CIPHER
3. VIGENERE CIPHER - *** Battista cipher
4. RAIL FENCE CIPHER

PIGPEN CIPHER



TEXT	PIGPEN SYMBOL
A	⌟
C	L
H	⊓
L	L·
W	V
Z	A·

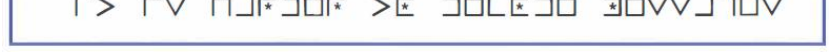
▲ Figure 3.13 The pigpen cipher

Activity 16

ACTIVITY 16

USING THE PIGPEN CIPHER

1 Encrypt the following message using the standard pigpen cipher:
I like encoding messages.
2 Decode the following message, which is written in the standard pigpen cipher:



IT IS HARDER TO DECODE MESSAGES

CAESAR CIPHER

Very simple encryption
Would not be used today
Esay to solve
Julius Caesar who encrypted message in the following way:
letters of the alphabet are shifted
positive – right
negative – left

PLAIN TEXT (INPUT)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
CIPHER TEXT (OUTPUT)	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B

▲ Table 3.4 The Caesar cipher with a key of –2

The Caesar cipher with a key of +3

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Positive (+3)	**Negative (-3)**
A = 1 A + 3 = 1 +3 =4 ➔ D H = 8 H+3=8+3=11 ➔ K HELLO ➔ KHOOR	D = 4 D – 3 = 4 – 3 = 1 ➔ A K = 11 K – 3 = 11 - 3 = 8 ➔ H KHOOR ➔ HELLO

Activity 17

ACTIVITY 17

USING THE CAESAR CIPHER

Encrypt the following using the Caesar cipher with a key of +3.

'THIS IS A MESSAGE.'

The Caesar cipher with a key of +3

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

```
THIS – WKLV
IS – LV
A – D
MESSAGE - PHVVDJH
‘THIS IS A MESSAGE’ = ‘WKLV LV D PHVVDJH’
```

Activity 18

ACTIVITY 18

CODING AN ENCRYPTION ALGORITHM

In the language you are studying, create and test a program to encrypt or decrypt a message using a key entered by a user.

```
#chapter15_activity18
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ'
LETTERS = LETTERS.lower()
def encrypt(message,key):
    encrypted=""
    for chars in message:
        if chars in LETTERS:
            num=LETTERS.find(chars)
            num+=key
            encrypted+=LETTERS[num]
    return encrypted
def decrypt(message,key):
    decrypted=""
    for chars in message:
        if chars in LETTERS:
            num=LETTERS.find(chars)
            num-=key
            decrypted+=LETTERS[num]
    return decrypted
def main():
    message=input("Enter your message:")
    key=int(input("Enter your key [1-26]:"))
    choice=input("Encrypt or Decrypt (e or d)?")
```

```
    if choice.lower().startswith('e'):
        print(encrypt(message,key))
    else:
        print(decrypt(message,key))
main()
```

VIGENERE CIPHER

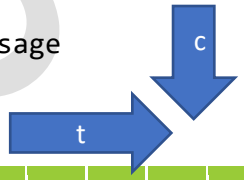
26 possible Caesar ciphers

So that each message will be encrypted differently, a **keyword** is chosen for each encryption. This is similar to the key chosen for the Caesar cipher, but there are far more, almost limitless options, making the cipher far more difficult to solve.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Vigenère Cipher Encrypted

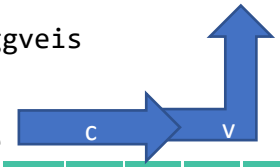
Plain text message - This is a message
Key stream - code
Row and column
plain text and keystream



Plain Text	t	h	i	s	i	s	a	m	e	s	s	a	g	e
Key stream	c	o	d	e	c	o	d	e	c	o	d	e	c	o
Cipher Text	v	v	l	w	k	g	d	q	g	g	v	e	i	s

Vigenère Cipher Decrypted

Plain text message - vvlwkgdqggveis
Key stream – code
Column and Row –
From code to decrypted message



Decrypted Message	v	v	l	w	k	g	d	q	g	g	v	e	i	s
Key stream	c	o	d	e	c	o	d	e	c	o	d	e	c	o
Text	t	h	i	s	i	s	a	m	e	s	s	a	g	e

RAIL FENCE CIPHER

Encrypt using Rail Fence Cipher

Plain text = This is a message
key = 3
This is a message =TIEGHSSMSAEIAS

T				**I**				**E**				**G**	
	H		**S**		**S**		**M**		**S**		**A**		**E**
		I				**A**				**S**			

Plain text = This is a message
key = 4
This is a message = TAGHSMAEIIESSS

T						**A**						**G**	
	H				**S**		**M**				**A**		**E**
		I		**I**				**E**		**S**			
			S						**S**				

Decrypt using Rail Fence Cipher

Cipher - TWLIIHSEBOGNWNENRSWNTEDAIIH
Key = 4

-						**-**					**-**					**-**						**-**		
	-				-		-				-		-				-		-				-	
		-		-				-		-				-		-				-		-		-
			-						-						-						-			

T						**W**						**L**						**I**						**I**		
	H				S		E				B		O				G		N				W		N	
		E		N				R		S				W		N				T		E				D
			A						I						I						H					

CHECKPOINTS ***

UNIT QUESTIONS ***