

UNIT 1: PROBLEM SOLVING

1 UNDERSTANDING ALGORITHMS

ACTIVITY 1

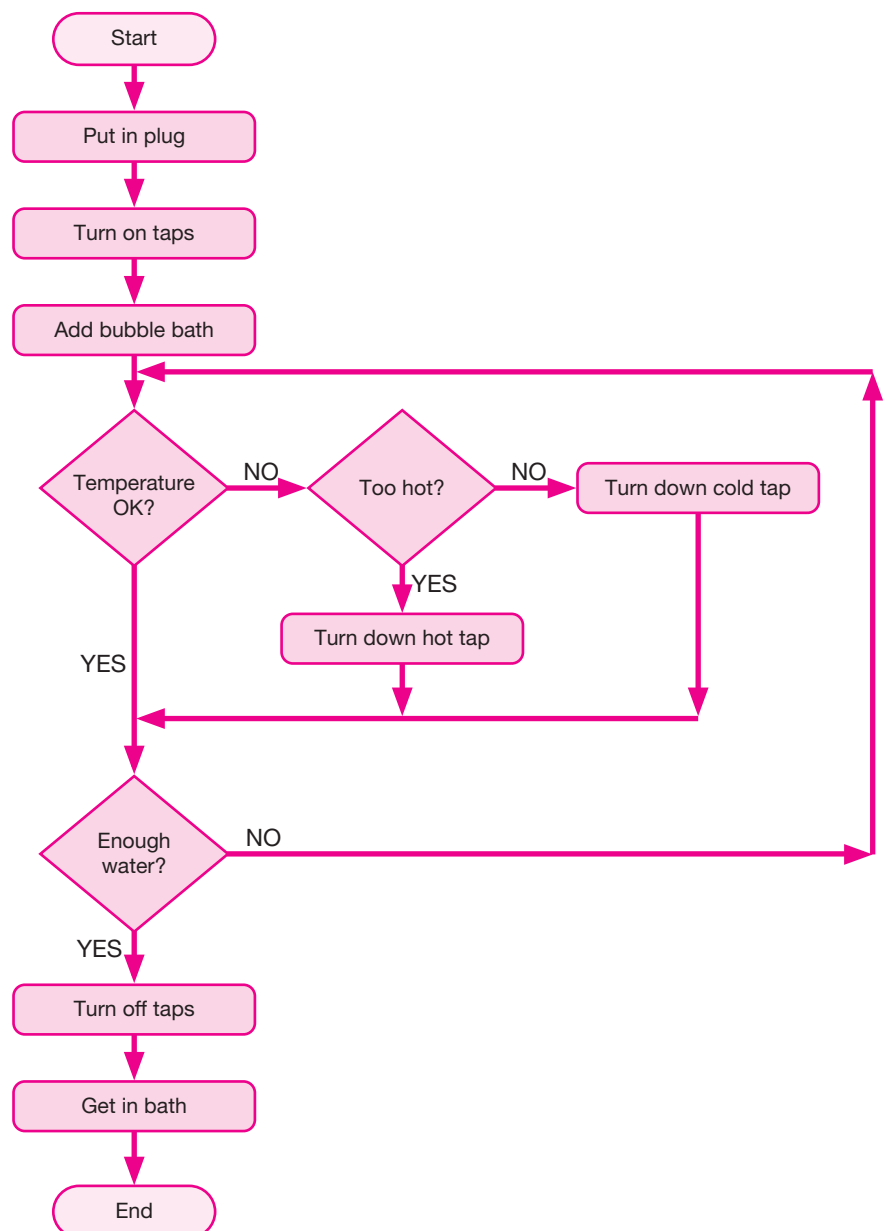
There is no single correct solution to this activity, since every student's walk to school will be unique. It is intended to highlight the need for instructions to be specific, precise and in a logical order. Encourage them to identify instances of repetition and selection, e.g. 'Keep walking until you get to the end of the street. Next cross the road, providing no cars are coming.'

ACTIVITY 2

Encourage students to use all the symbols shown in Figure 1.2 in their flowchart.

ACTIVITY 3

This flowchart incorporates selection and iteration. Students may simply identify a sequence of processes in their solution.



ACTIVITY 4

Students must be familiar with Pearson Edexcel pseudocode, which will be used to frame some of the exam questions on Paper 2, but they don't have to use it themselves.

ACTIVITY 5

```

SEND 'Enter first number.' TO DISPLAY
RECEIVE firstNumb FROM KEYBOARD
SEND 'Enter second number.' TO DISPLAY
RECEIVE secondNumb FROM KEYBOARD
SET thirdNumb TO firstNumb * secondNumb
SEND thirdNumb TO DISPLAY

```

ACTIVITY 6

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.

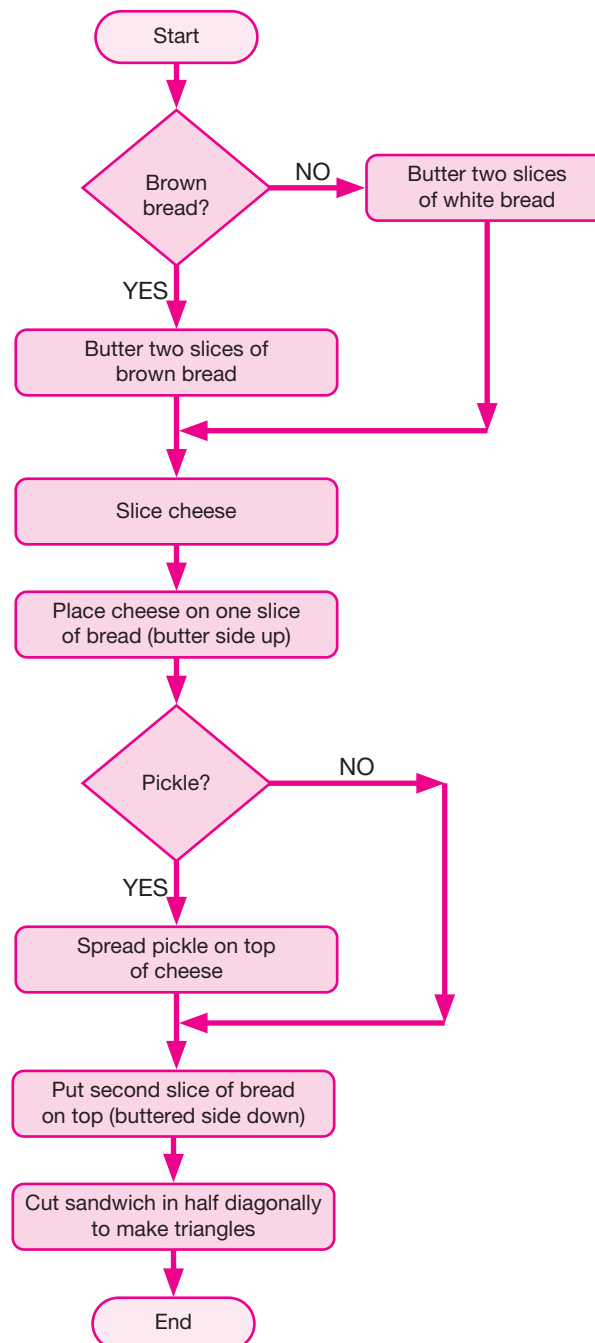
CHECKPOINT

S1 The specifics of this algorithm should reflect the procedure for borrowing books at the student's own local library or school library.

S2 Table 1.1 describes the function of each of the arithmetic operators.

S3 and S4 Variables and constants are explained on page 9.

C1



C2

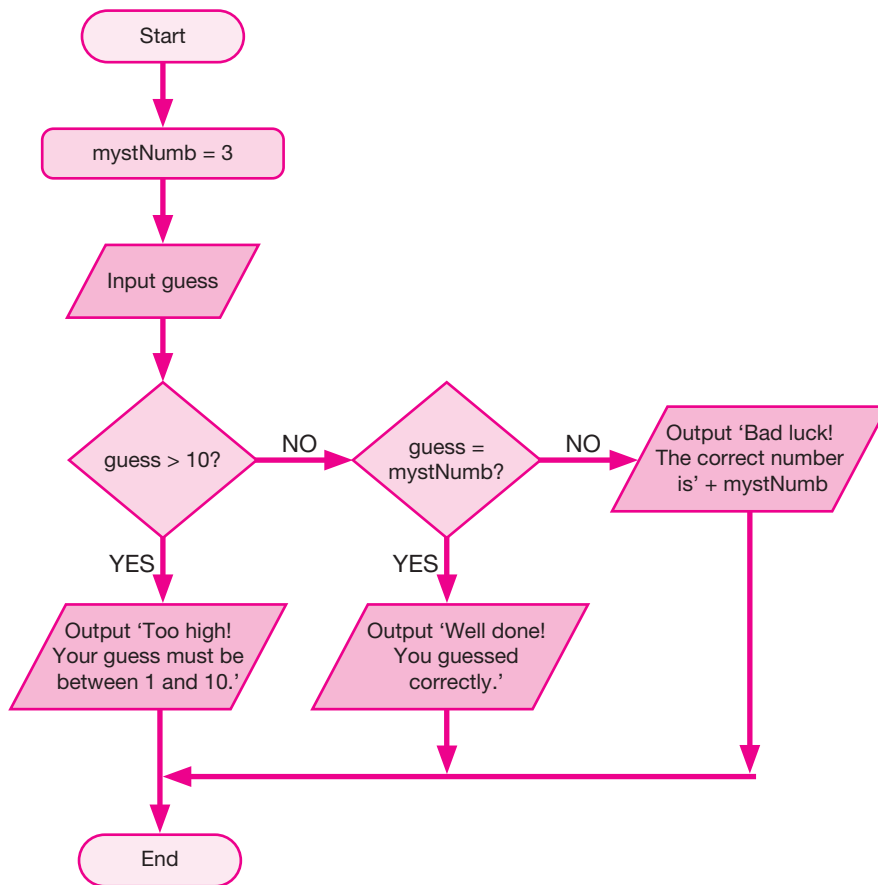
```

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

```

2 CREATING ALGORITHMS**ACTIVITY 7**

Here's the algorithm expressed as a flowchart.



Here's the same algorithm expressed as pseudocode.

```

SET mystNumb TO 3
SEND 'Please enter a number between 1 and 10.' TO DISPLAY
RECEIVE guess FROM KEYBOARD
IF guess > 10 THEN
    SEND 'Too high! Your guess must be between 1 and 10.' TO DISPLAY

```

```
ELSE
  IF guess = mystNumb THEN
    SEND 'Well done! You have guessed correctly.' TO DISPLAY
  ELSE
    SEND 'Bad luck! The correct number is' & mystNumb TO DISPLAY
  END IF
END IF
```

ACTIVITY 8

SCORE	OUTPUT
91	A
56	D
78	B

CHECKPOINT

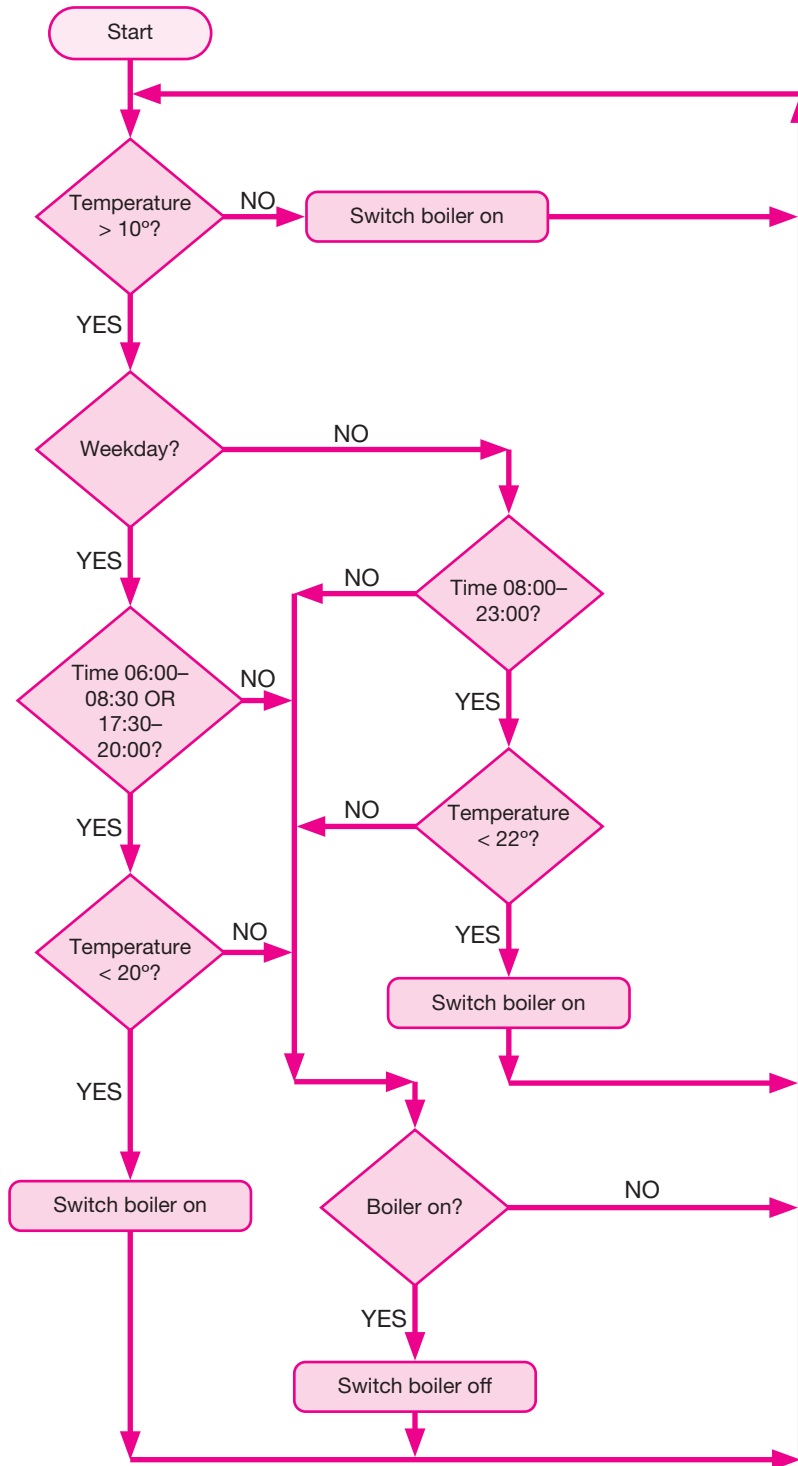
S1 The activities in this chapter provide plenty of examples that students can use to illustrate their answer.

C1

```
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 body mass index (BMI) is:' & weight/height^2 TO DISPLAY
```

(Reminder: height^2 means 'height to the power of 2', i.e. height^2 . See 'Arithmetic operators' on page 9 of the Student Book.)

C2 Encourage students to devise appropriate test data to check that the algorithm produces the right outcomes.



3 SORTING AND SEARCHING ALGORITHMS

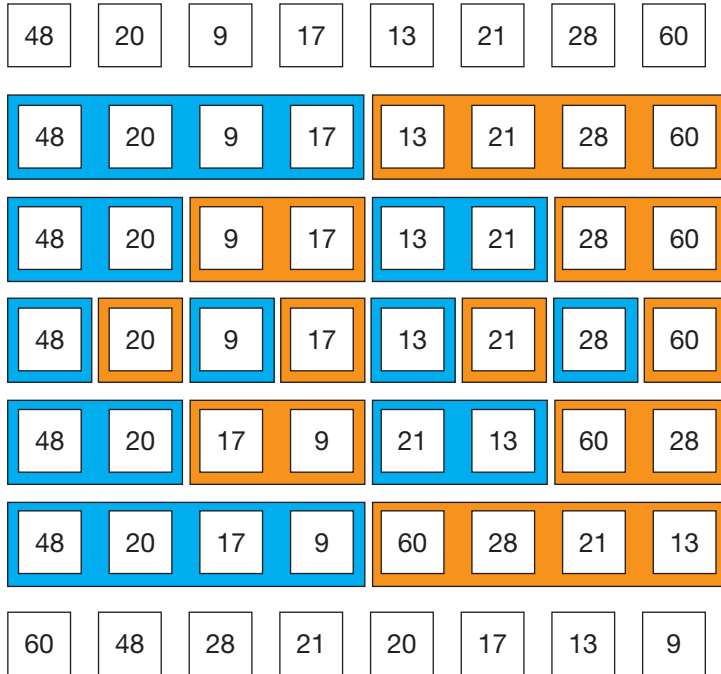
ACTIVITY 9

The variable `length` is used to store the length of the list. 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. 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.

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. The algorithm then terminates.

ACTIVITY 10



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

CHECKPOINT

S1 The difference between the bubble sort and merge sort algorithms is described on page 19.

S2 The way in which a binary search algorithm finds the search item is explained on page 20.

C1 The efficiency of a linear search as compared to a binary search is described on page 21, which includes mention of when it might be better to use the former rather than the latter.

4 DECOMPOSITION AND ABSTRACTION

ACTIVITY 12

Inputs:

- when to start a new game
- number and names of players
- when to spin the wheel
- a player's selected answer
- whether players want to play again.

Outputs:

- A message to inform a player when it is their turn.
- A message to inform the player of the outcome of spinning the wheel (question category).
- A question plus four possible answers.
- A message to inform the player whether their answer is correct or incorrect.

- A message to inform the player that they can have another go.
- A message to inform the player how many points they have scored.
- A message at the end of each round to inform each player of their total score.
- A 'game over' message.
- A message at the end of the game informing players who has won.
- A message to ask whether the players want to play another game or want to finish.

Processing requirements:

- Set up question banks for each colour/subject.
- Flag each question as unused.
- Establish how many players there are (up to a maximum of four).
- Set each player's score to 0.
- Repeat until one player reaches a score of at least 30 or there are no more unanswered questions.
- Prompt next player to select a subject and simulate spinning the wheel. Display colour and subject selected.
- Randomly select a question from the remaining unanswered questions in the subject question bank.
- Display the selected question and four possible answers. Prompt player to select an answer.
- 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.

Display an appropriate message.

- Flag the question as used.
- When one of the players reaches a score of 30 or more or the computer runs out of questions stop the game.
- If there is a winner, display name and congratulatory message. If the computer has run out of questions, display an appropriate message.
- Check if the players want to play another game or finish.

Subprograms:

- select_category
- display_Q&A
- check_response
- update_score
- mark_asked_questions
- establish_winner

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.

Algorithm in 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 colour selected is:' & colour TO DISPLAY

```

CHECKPOINT

S1 Decomposition is described on pages 23–24.

S2 Abstraction is described on pages 24–25.

S3 Abstractions would include distance of journey; cost of fuel; fuel usage of vehicle; distance / fuel usage.

C1 Students could use their solution to Activity 12 to illustrate the use of decomposition and abstraction. Alternatively, they may wish to come back to this checkpoint once they have completed the chapter.

C2 Computational thinking is defined in the Subject vocabulary box on page 23.

C3 We use abstractions when we refer to objects or concepts. For example, we say ‘car’ and might expect everyone to have an image of a small vehicle with four wheels. We do not have to describe it in full detail.

UNIT QUESTIONS

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, customer, age.

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

4 $(2 \times £5) + (2 \times £10) + £9 = £39$ (They are not eligible)

5

Marek, Jackson, Bachchan, Wilson, Abraham, French, Smith
 Jackson, Bachchan, Marek, Abraham, French, Smith, Wilson
 Bachchan, Jackson, Abraham, French, Marek, Smith, Wilson
 Bachchan, Abraham, French, Jackson, Marek, Smith, Wilson
 Abraham, Bachchan, French, Jackson, Marek, Smith, Wilson

6

Azikiwe, Bloom, Byrne, Davidson, Gateri, Hinton, Jackson, Linton, Smith, Wall

Median is Hinton: Jackson > Hinton

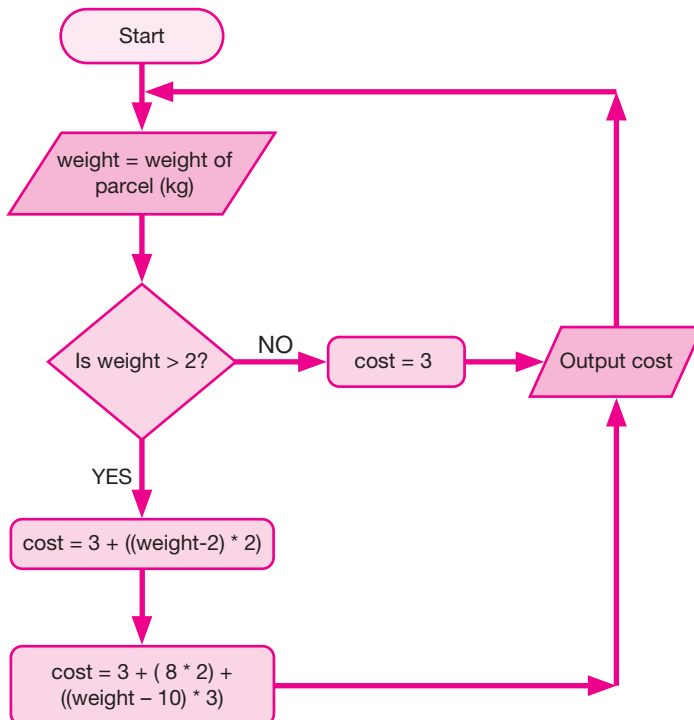
First half of list discarded. New list: Hinton, Jackson, Linton, Smith, Wall

Median is Linton: Jackson < Linton

Second half of list discarded. New list: Hinton, Jackson

Median is Jackson: search item found

7 a



b

```

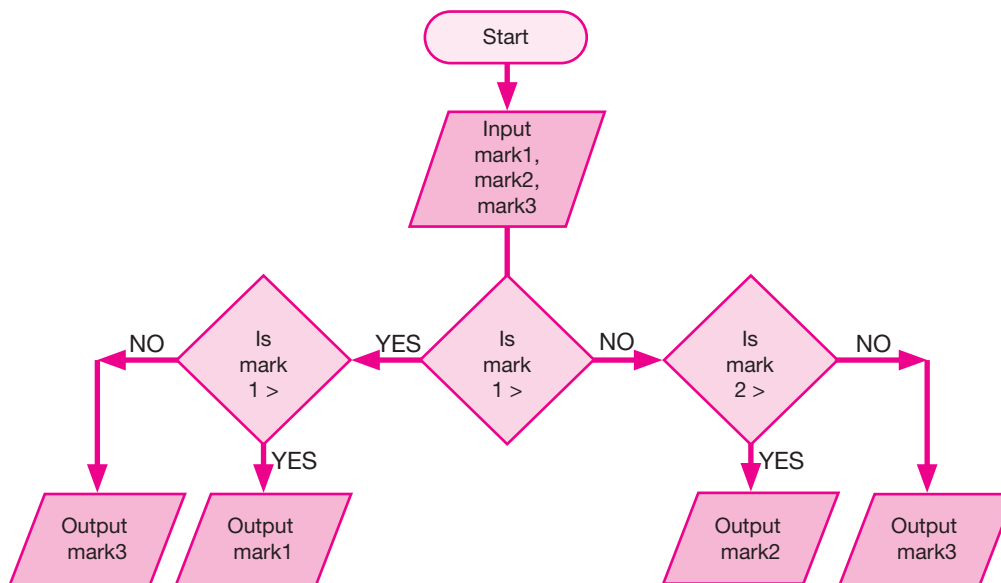
SET parcel TO "y"
WHILE parcel = "y" DO
    RECEIVE weight FROM (INTEGER) KEYBOARD
    IF weight <= 2 THEN
        SET cost TO 3
        IF weight > 10 THEN
            SET cost TO 3 + (8*2) + ((weight - 10) * 3))
        ELSE
            SET cost TO 3 + ((weight - 2)*2)
        END IF
    END IF
    SEND cost TO PRINTER
    SEND 'Press "y" to process another parcel.' TO DISPLAY
    RECEIVE parcel FROM (STRING) KEYBOARD
END WHILE

```

The loop will run while parcel = "y"

Parcels up to 2kg cost £2

8



9

Pass 1

1 4 2 6 3 5
1 2 4 6 3 5
1 2 4 6 3 5
1 2 4 3 6 5
1 2 4 3 5 6

Pass 2

1 2 4 3 5 6
1 2 4 3 5 6
1 2 3 4 5 6

UNIT 2: PROGRAMMING

5 DEVELOP CODE

ACTIVITY 1

- 1 Students should create a table listing data types. High-level programming languages often provide more than the four basic data types.
- 2
 - a integer
 - b real
 - c Boolean
- 3 There is no single correct solution to this activity.

ACTIVITY 2

VARIABLE	PURPOSE	DATA TYPE
totalVisitors	Keeps track of the number of people in the park at any one time. (It is incremented by one each time someone enters and decremented by 1 each time someone leaves.)	integer
visitorType	Set to 'a' if the visitor is an adult and 'c' if the visitor is a child.	character
Taking	Keeps a running total of the amount of money collected at the gate (AED 125 per adult; no charge for children).	real
parkFull	Set to 'False' initially but changes to 'True' when the number of visitors reaches 10,000.	Boolean

ACTIVITY 3

- 1 The algorithm takes in two integer numbers entered from the keyboard and displays:
 - the result of dividing the first number by the second number
 - the number remaining after dividing the first number by the second number
 - the integer division.
- 2
 - a With 4 and 2 as inputs, the outputs would be: 2.0, 0, 2.
 - b With 10 and 3 as inputs, the outputs would be: 3.33333333333333, 1, 3.
 - c With 20 and 6 as inputs, the outputs would be: 3.33333333333333, 2, 3.
- 3 Here is the algorithm implemented in Python:

```
number1 = int(input('Enter first number:'))
number2 = int(input('Enter second number:'))
print('number1 / number2 =', number1/number2)
print('number1 MOD number2 =', number1 % number2)
print('number1 DIV number2 =', number1 // number2)
```

Here is the algorithm implemented in C#.

```
int number1, number2;
string number1String, number2String;
```

```

Console.WriteLine("Enter first number:");
number1String = Console.ReadLine();
number1 = int.Parse(number1String);

Console.WriteLine("Enter second number:");
number2String = Console.ReadLine();
number2 = int.Parse(number2String);

Console.WriteLine("number1 / number2 =" + (float)number1 / (float)number2);
Console.WriteLine("number1 MOD number2 =" + number1 % number2); // in C# / defaults to
integer division
Console.WriteLine("number1 DIV number2 =" + number1 / number2);

```

Here is the algorithm implemented in Java:

```

import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("Enter first number:");
        int number1 = scan.nextInt();

        System.out.println("Enter second number:");
        int number2 = scan.nextInt();
        scan.close();

        System.out.printf("number1 / number2 = %f\n", (float) number1 / (float) number2);
        System.out.printf("number1 MOD number2 = %d\n", number1 % number2);
        System.out.printf("number1 DIV number2 = %d\n", number1 / number2);
    }
}

```

ACTIVITY 4

SCORE	HIGHSCORE	OUTPUT
5	10	You haven't beaten your high score.
20	10	You've exceeded your high score!
15	15	You haven't beaten your high score.

ACTIVITY 5

Here is the algorithm implemented in Python:

```

age = int(input('Please enter your age:'))

if age <= 18:
    numLessons = 20

```

```

else:
    numLessons = 20 + (age - 18) * 2
print('You need', numLessons, 'driving lessons.')

```

Here is the algorithm implemented in C#:

```

int age, numLessons;
string ageString;

Console.WriteLine("Please enter your age:");
ageString = Console.ReadLine();
age = int.Parse(ageString);

if (age <= 18)
{
    numLessons = 20;
}
else
{
    numLessons = 20 + (age - 18) * 2;
}

Console.WriteLine("You need" + numLessons + "driving lessons.");

```

Here is the algorithm implemented in Java:

```

import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Please enter your age:");
        int age = scan.nextInt();
        scan.close();

        int numLessons = 20 + (age - 18) * 2;
        if(age <=18) {
            numLessons = 20;
        }
        System.out.printf("You need %d driving lessons.\n", numLessons);
    }
}

```

ACTIVITY 6

Python:

```

start = int(input('Enter the start number'))
end = int(input('Enter the end number'))
total = 0

for count in range(start, end + 1):
    total = total + count
print(total)

```

Java:

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the start number:");
        int start = scanner.nextInt();

        System.out.print("Enter the end number:");
        int end = scanner.nextInt();
        scanner.close();

        int total = 0;
        for(int count = start; count < end; count++) {
            total += count;
        }
        System.out.println(total);
    }
}
```

C#:

```
int startNumber, endNumber, total;
    string startNumberString, endNumberString;

    Console.WriteLine("Enter start number:");
    startNumberString = Console.ReadLine();
    startNumber = int.Parse(startNumberString);

    Console.WriteLine("Enter end number:");
    endNumberString = Console.ReadLine();
    endNumber = int.Parse(endNumberString);

    total = 0;

    for (int x = startNumber; x <= endNumber; x++)
    {
        total += x;
    }

    Console.WriteLine("Total is" + total);
```

ACTIVITY 7

- a** It uses nested loops to calculate the average mark for 5 pieces of work from each of twenty students.
- b** To ensure that the marks entered by the user are recognised as integers and not strings.

ACTIVITY 8

Python:

```
for table in range(2, 13):
    for times in range(2, 13):
        print(str(times) + 'x' + str(table) + '=' + str(table * times))
```

Java:

```
class Main {
    public static void main(String[] args) {
        for(int table = 2; table < 13; table++) {
            for(int times = 2; times < 13; times++) {
                int answer = table * times;
                System.out.println(times + "x" + table + "=" + answer);
            }
        }
    }
}
```

C#:

//iterates through required multipliers

```
for (int multiplier = 2; multiplier <= 12; multiplier++)
{
    Console.WriteLine("The" + multiplier + "times table:");
```

//display times table

```
for (int x = 1; x <= 12; x++)
{
    Console.WriteLine(x + "x" + multiplier + "=" + (x * multiplier));
}
```

// add blank line between each times table

```
    Console.WriteLine("");
}
```

ACTIVITY 9

1 a Algorithm A displays the numbers 1 to 10 cubed, i.e. 1–1000.

Here is the algorithm implemented in Python:

```
for index in range(1, 11):
    print (index * index * index)
```

Here is the algorithm implemented in C#:

```
for (int i = 1; i <= 10; i ++ )
{
    Console.WriteLine(i * i * i);
}
```

Here is the algorithm implemented in Java:

```
class Main {
    public static void main(String[] args) {
        for(int i = 1; i < 11; i++) {
            System.out.println(i * i * i);
        }
    }
}
```

- b** Algorithm B displays a countdown from 10 to 1. Here is the algorithm implemented in Python:

```
counter = 10
while counter > 0:
    print(counter)
    counter -= 1
```

Here is the algorithm implemented in C#:

```
int counter = 10;

while (counter > 0)
{
    Console.WriteLine(counter);
    counter--;
}
```

Here is the algorithm implemented in Java:

```
class Main {
    public static void main(String[] args) {
        int counter = 10;
        while(counter > 0) {
            System.out.println(counter);
            counter--;
        }
    }
}
```

ACTIVITY 10

Python:

```
import random
play = True
correct = False

while play == True:
    randomNumber = random.randint(1, 21)
    while correct == False:
        guess = int(input('Please enter a number between 1 and 20:'))
        if guess == randomNumber:
```

```

        print('Your guess is correct.')
        correct = True
    elif guess < randomNumber:
        print('Your guess is too low.')
    elif guess > randomNumber:
        print('Your guess is too high.')
    reply = (input('Do you want to play again? (y or n)'))
    reply = reply.upper()
    if reply == 'Y':
        play = True
        correct = False
    else:
        play = False

```

Java:

```

import java.util.*;
class Main {
    public static void main(String[] args) {
        Random r = new Random();
        Scanner scanner = new Scanner(System.in);

        // choose a random number between 1 and 20
        int randomNumber = r.nextInt(20) + 1;
        boolean askAgain = true;
        // keep asking the user to guess until they get it wrong or enter NO
        while (askAgain) {
            System.out.print("Guess a number between 1 and 20:");
            // read a string so that it could be NO as well as a number
            String guess = scanner.nextLine();
            // stop asking if the user types in NO
            if(guess == "NO") {
                break;
            }
            // convert guess from string to integer
            int intGuess = Integer.parseInt(guess);
            // check guess and feedback to user
            if(intGuess > randomNumber) {
                System.out.println("Too high - try again");
            } else if (intGuess < randomNumber) {
                System.out.println("Too low - try again");
            } else {
                System.out.println("Correct: Well done!");
                askAgain = false;
            }
        }
    }
}

```


C#:

```

string guessString;
    string playAgain;
    int guess = 0;
    int randomNumber;

    do
    {
        //generate random number
        Random rand = new Random();
        randomNumber = rand.Next(1, 21);
        Console.WriteLine(randomNumber);

        guess = 0;
        while (guess != randomNumber)
        {
            Console.WriteLine("Please enter your guess");
            guessString = Console.ReadLine();
            guess = int.Parse(guessString);
            if (guess == randomNumber)
            {
                Console.WriteLine("Correct");
            }
            else if (guess < randomNumber)
            {
                Console.WriteLine("Too low");
            }
            else
            {
                Console.WriteLine("Too high");
            }
        }
        Console.WriteLine("Do you want to play again? NO quits.");
        playAgain = Console.ReadLine();
    } while (playAgain != "NO");

```

CHECKPOINT

- S1** Variables are defined in the subject vocabulary box on page 8 of Unit 1 and page 9 includes an explanation of the purpose of variables. Students need to understand that variables have a variety of uses, for example controlling the number of times a loop is executed, determining which branch of an **IF** statement is taken, keeping running totals and holding user input, etc.
- S2** Ideally, students should give examples of different data types from their own programs.
- S3** The description will vary according to the high-level language the student is studying. The main thing for them to realise is that the language will have a number of different selection and loop constructs.
- C1** Command sequences, selection and iteration were described in detail in Unit 1. They are revisited in this unit on pages 38–49. Variable initialisation is covered on pages 35–36. As well as being able to describe these constructs, students should also be able to recognise them in an algorithm.

6 MAKING PROGRAMS EASY TO READ

ACTIVITY 11

1

SET countDown TO 10	#The loop counts down from 10 to 0
WHILE countDown >= 0 DO	
IF countDown > 0 THEN	
SEND countDown TO DISPLAY	
ELSE	
SEND 'Blast Off' TO DISPLAY	#Prints 'Blast Off' when 0 is reached
END IF	
SET countDown to countDown - 1	#Ensures that the loop will terminate
END WHILE	

Here is the algorithm implemented in Python:

```
countDown = 10
while countDown >= 0:
    if countDown > 0:
        print(countDown)
    else:
        print('Blast off')
    countDown -= 1
```

Here is the algorithm implemented in C#:

```
int countDown = 10;

while (countDown >= 0)
{
    if (countDown > 0)
    {
        Console.WriteLine(countDown);
    }
    else
    {
        Console.WriteLine("Blast off");
    }
    countDown -= 1;
}
```

Here is the algorithm implemented in Java:

```
class Main {
    public static void main(String[] args) {
        int countDown = 10;
        while(countDown >= 0) {
            if(countDown > 0) {
                System.out.printf("%d\n", countDown);
            }
            countDown--;
        }
    }
}
```

```

    } else {
        System.out.println("Blast Off");
    }
    countDown -= 1;
}
}
}

```

2 Here is the calculator algorithm expressed as source code:

REPEAT

REPEAT

SEND 'Select an option: a - addition, s - subtraction, d - division or m - multiplication' TO DISPLAY

RECEIVE choice FROM KEYBOARD

UNTIL choice = 'a' OR choice = 's' OR choice = 'd' OR choice = 'm'

SEND 'Enter the first number' TO DISPLAY

RECEIVE number1 FROM KEYBOARD

SEND 'Enter the second number' TO DISPLAY

RECEIVE number2 FROM KEYBOARD

IF choice = 'a' THEN

SEND number1 + number2 TO DISPLAY

ELSE

IF choice = 's' THEN

SEND number1 - number2 TO DISPLAY

ELSE

IF choice = 'd' THEN

SEND number1 / number2 TO DISPLAY

ELSE

SEND number1 * number2 TO DISPLAY

END IF

END IF

END IF

SEND 'Another go?' TO DISPLAY

RECEIVE anotherGo FROM KEYBOARD

UNTIL anotherGo <> 'y' OR anotherGo <> 'Y'

Here is the algorithm implemented in Python:

```
anotherGo = 'y'
```

```
while anotherGo == 'y' or anotherGo == 'Y':
```

```
while True:
```

#Loop to filter out invalid choices

```
choice = input('Select an option; a - addition, s - subtraction, d - division or m - multiplication.')
```

```

if choice == 'a' or choice == 's' or choice == 'd' or choice == 'm':
    break
else:
    print('Invalid selection.')

firstNumber = int(input('Enter the first number.'))
secondNumber = int(input('Enter the second number.'))

if choice == 'a':
    print(firstNumber + secondNumber)
elif choice == 's':
    print(firstNumber - secondNumber)
elif choice == 'd':
    print(firstNumber / secondNumber)
else:
    print(firstNumber * secondNumber)

anotherGo = input('Another go?')

```

Here is the algorithm implemented in C#:

```

bool acceptableInput;
string choice = "", anotherGo = "y";
string firstNumberString, secondNumberString;
int firstNumber, secondNumber;

while (anotherGo == "y" || anotherGo == "Y")
{
    acceptableInput = false;
    // validate input is a, s, d or m
    while (acceptableInput == false)
    {
        Console.WriteLine("Select an option: a - addition, s - subtraction, d
        - division or m - multiplication");
        choice = Console.ReadLine();

        if (choice == "a" || choice == "s" || choice == "d" || choice == "m")
        {
            acceptableInput = true;
        }
        else
        {
            Console.WriteLine("Invalid Choice");
        }
    }

    // get values from user
    Console.WriteLine("Enter the first number:");

```

```

firstNumberString = Console.ReadLine();
firstNumber = int.Parse(firstNumberString);           // convert string to integer
Console.WriteLine("Enter the second number:");
secondNumberString = Console.ReadLine();
secondNumber = int.Parse(secondNumberString);         // convert string to integer

```

```

if (choice == "a")
{
    Console.WriteLine(firstNumber + secondNumber);
}
else if (choice == "s")
{
    Console.WriteLine(firstNumber - secondNumber);
}
else if (choice == "d")
{
    Console.WriteLine(firstNumber / secondNumber);
}
else
{
    Console.WriteLine(firstNumber * secondNumber);
}
Console.WriteLine("Enter y for another go, any other input will quit");
anotherGo = Console.ReadLine();
}

```

Here is the algorithm implemented in Java:

```

import java.util.*;

class Main {
    public static void main(String[] args) {
        String anotherGo = "y";
        Scanner scan = new Scanner(System.in);

        while(anotherGo.equalsIgnoreCase("y")) {
            String choice;
            while(true) {                                     // loop to filter out invalid choices
                System.out.println("Select an option: a - addition, s - subtraction, d -
                division or m - multiplication.");
                choice = scan.nextLine();
                if(choice.equals("a") || choice.equals("s") || choice.equals("d") || choice.
                equals("m")) {
                    break;
                } else {
                    System.out.println("Invalid selection");
                }
            }
        }
    }
}

```

```

System.out.print("Enter the first number.");
int firstNumber = scan.nextInt();
scan.nextLine(); // read newline after number
System.out.print("Enter the second number.");
int secondNumber = scan.nextInt();
scan.nextLine(); // read newline after number

```

```

switch(choice) {
    case "a":
        System.out.printf("%d \n", firstNumber + secondNumber);
        break;
    case "s":
        System.out.printf("%d \n", firstNumber - secondNumber);
        break;
    case "d":
        System.out.printf("%f \n", (float)firstNumber / (float)secondNumber);
        break;
    case "m":
        System.out.printf("%d \n", firstNumber * secondNumber);
        break;
}
System.out.println("Another go?");
anotherGo = scan.nextLine();
}
scan.close();
}
}

```

CHECKPOINT

S1 Covered on pages 51–52.

S2 Listed in Table 2.5 on page 52.

C1 Solution not required. Encourage students to look at each other's code and assess its readability.

7 STRINGS

ACTIVITY 12

```

SEND 'Enter your password.' TO DISPLAY
RECEIVE password FROM KEYBOARD
IF LENGTH(password) < 6 THEN
    SEND 'The password you have entered is not long enough.' TO DISPLAY
ELSE
    SEND 'Length of password OK.' TO DISPLAY
END IF

```

Here is the algorithm implemented in Python:

```

password = input('Enter your password.')
if len(password) < 6:
    print('The password you have entered is not long enough.')

```

```
else:
    print('Length of password OK.')
```

Here is the algorithm implemented in C#:

```
string password;

    Console.WriteLine("Enter your password.");
    password = Console.ReadLine();

    if (password.Length < 6)
    {
        Console.WriteLine("The password you have entered is not long enough.");
    }
    else
    {
        Console.WriteLine("Length of password OK.");
    }
    Console.ReadLine();
```

Here is the algorithm implemented in Java:

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        // enter password
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a password:");
        String password = scanner.nextLine();

        // check password length
        if(password.length() < 6) {
            System.out.println("The password you have entered is not long
enough");
        } else {
            System.out.println("Length of password OK");
        }
    }
}
```

ACTIVITY 13

Python:

```
string = 'The cars present include Ford, Mercedes, Toyota, BMW, Audi and Renault'

make = input('Please enter a make to search for.')
make = make.upper()
string = string.upper()
present = make in string
```

```

if present == True:
    print('It is present')
else:
    print('It is not present')

```

Java:

```

import java.util.*;
class Main {
    public static void main(String[] args) {
        String cars = "The cars present include Ford, Mercedes, Toyota, BMW, Audi
and Renault";
        Scanner scanner = new Scanner(System.in);
        System.out.print("Please enter a make to search for:");
        String make = scanner.nextLine();

        // convert both to upper case
        cars = cars.toUpperCase();
        make = make.toUpperCase();

        // check if cars contains the string that the user entered
        if(cars.contains(make)) {
            System.out.println("It is present");
        } else {
            System.out.println("It is not present");
        }
    }
}

```

C#:

Note: Many, more efficient solutions to this exist, but this simple method may be more likely to be used by beginner programmers. Other solutions may use `Array.Exists` or `import LINQ`.

```

string[] make = {"Ford", "Mercedes", "Toyota", "BMW", "Audi", "Renault"};
string userChoice;
bool present = false;

```

```

Console.WriteLine("Please enter a make:");
userChoice = Console.ReadLine();

```

// go through each element in the array and check if make is present

```

foreach (string currentElement in make)
{
    if (currentElement.ToLower() == userChoice.ToLower())
    {
        present = true;
    }
}

if (present == true)
{
    Console.WriteLine("It is present");
}

```



```

}
else
{
    Console.WriteLine("It is not present");
}

```

ACTIVITY 14

Here is the program implemented in Python:

```

firstName = input('Enter your first name:')
surname = input('Enter your surname:')
length = len(surname)

```

#Deals with surnames less than 4 characters long

```

if length == 1:
    surname = surname & 'xxx'
elif length == 2:
    surname = surname & 'xx'
elif length == 3:
    surname = surname & 'x'

first4Letters = surname[0:4]
userName = firstName[0] & first4Letters
print('Your username is:', userName)

```

Here is the algorithm implemented in C#:

```

string firstName, surname, username;

    Console.WriteLine("Enter your first name:");
    firstName = Console.ReadLine();
    Console.WriteLine("Enter your surname name:");
    surname = Console.ReadLine();

    // deal with surnames less than 4 characters long by adding 3 XXXs then just use first four characters
    surname = surname + "XXX";

    username = firstName.Substring(0, 1) + surname.Substring(0, 4);
    Console.WriteLine("Your username is:" + username);

```

Here is the algorithm implemented in Java:

```

import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.println("Enter your first name:");
        String firstName = scan.nextLine();
        System.out.println("Enter your surname:");
        String surname = scan.nextLine();
    }
}

```

```

        // Deals with surnames less than 4 characters long
        while(surname.length() < 4) {
            surname = surname + "X";
        }

        String username = firstName.substring(0, 1) + surname.substring(0, 4);

        System.out.printf("Your username is: %s\n", username);
        scan.close();
    }
}

```

CHECKPOINT

S1 String indexing is explained on page 54.

S2 The program students developed in Activity 13 uses a loop to traverse a string.

S3 Concatenation and slicing strings are explained on pages 59–60.

C1 Here is the program implemented in Python:

```

sentence = input('Enter a sentence. ')
separateStrings = sentence.split(' ')           #Splits up the sentence into a list of strings
for item in separateStrings:                   #Prints each item in separateStrings on a separate line
    print(item)

```

Here is the algorithm implemented in C#:

```

Console.WriteLine("Enter a sentence.");
    string sentence = Console.ReadLine();
    string[] separateStrings = sentence.Split(' ');
    foreach (string word in separateStrings)
    {
        Console.WriteLine(word);
    }

```

Here is the algorithm implemented in Java:

```

import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Enter a sentence.");
        String sentence = scan.nextLine();

        String separateStrings[] = sentence.split(""); // Splits up the sentence into an array of strings
        for(int i = 0; i < separateStrings.length; i++) {
            System.out.println(separateStrings[i]);
        }

        scan.close();
    }
}

```

8 DATA STRUCTURES

ACTIVITY 15

Python:

```
cars = ['Ford', 'Mercedes', 'Toyota', 'BMW', 'Audi', 'Renault']
length = len(cars)
for name in range(0, length):
    print(cars[name])
```

Java:

```
class Main {
    public static void main(String[] args) {
        // create an array of strings
        String[] cars = {"Ford", "Mercedes", "Toyota", "BMW", "Audi", "Renault"};

        // find the length
        int length = cars.length;

        // use a for loop to traverse the array
        for(int i = 0; i < length; i++) {
            System.out.println(cars[i]);
        }
    }
}
```

C#:

```
string[] make = {"Ford", "Mercedes", "Toyota", "BMW", "Audi", "Renault"};
Console.WriteLine("Number of elements:" + make.Length);
foreach (string currentElement in make)
{
    Console.WriteLine(currentElement);
}
```

ACTIVITY 16

Python:

```
firstNames = ['Alex', 'Bryn', 'Eloise', 'Lois', 'James', 'Sally']
searchName = input('What name are you looking for?')
found = False
index = 0

while found == False and index <= (len(firstNames) - 1):
    if searchName == firstNames[index]:
        found = True
    else:
        index += 1
```

```

if found == True:
    print(searchName, 'is in position', index, 'in the list.')
else:
    print(searchName, 'is not in the list.')

```

Java:

```

import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // create list of names
        String[] firstNames = {"Alex", "Bryn", "Eloise", "Lois", "James", "Sally"};
        // ask which name to search for
        System.out.print("Name to search for:");
        String searchName = scanner.nextLine();
        scanner.close();
        boolean found = false;
        int index = 0;
        // linear search
        while (found == false && index < firstNames.length) {
            if(searchName.equals(firstNames[index])) {
                found = true;
            } else {
                index++;
            }
        }
        // display results
        if(found) {
            System.out.println(searchName + "is at position" + index + "in the list");
        } else {
            System.out.println(searchName + "is not in the list");
        }
    }
}

```

C#:

```

string[] firstNames = {"Alex", "Bryn", "Eloise", "Lois", "James", "Sally"};
string searchName;
bool found = false;
int index = 0;

Console.WriteLine("Enter name to search for:");
searchName = Console.ReadLine();

while (found == false && index <= firstNames.Length - 1)
{

```

```

        if (searchName == firstNames[index])
        {
            found = true;
            break;
        }
        index += 1;
    }

    if (found == true)
    {
        Console.WriteLine(searchName + "is at position" + index + "in the list");
    }
    else
    {
        Console.WriteLine(searchName + "is not in the list");
    }
}

```

ACTIVITY 17

Here is the program implemented in Python:

```

marks = [[80, 59, 34, 89], [31, 11, 47, 64], [29, 56, 13, 91], [55, 61, 48, 0],
[75, 78, 81, 91]]
highestMark = marks[0][0]
lowestMark = marks[0][0]
total = 0
count = 0
for row in marks:
    for column in row:
        total += column
        count +=1
        if column > highestMark:
            highestMark = column
        elif column < lowestMark:
            lowestMark = column
print('The highest mark is', highestMark)
print('The lowest mark is', lowestMark)
print('The average mark is', total/count)

```

#Initialises the list

#Initialises highest and lowest mark to first mark in the list

#Adds up the marks

#Counts the numbers of marks

Here is the algorithm implemented in C#:

```

int[,] marks = new int[,]
{
    {89,34,59,80},
    {64,47,11,31},
    {91,13,56,29},
    {0,48,61,55},
    {91,81,78,75}
};

```

```

    int highestMark, lowestMark, total, count;
    highestMark = marks[0, 0];
    lowestMark = marks[0, 0];
    total = 0;
    count = 0;

    foreach (int value in marks)
    {
        total += value;
        count++;

        if (value > highestMark)
        {
            highestMark = value;
        }
        else if (value < lowestMark)
        {
            lowestMark = value;
        }
    }

    Console.WriteLine("Highest mark is" + highestMark);
    Console.WriteLine("Lowest mark is" + lowestMark);
    Console.WriteLine("Average mark is" + total / count);

```

Here is the algorithm implemented in Java:

```

public class Main {
    public static void main(String[] args) {
        int[][] marks = {{80, 59, 34, 89}, {31, 11, 47, 64}, {29, 56, 13, 91}, {55, 61, 48, 0},
                        {75, 78, 81, 91}};

                                                                    //Initialises the list

        int highestMark = marks[0][0];
        int lowestMark = marks[0][0];

                                                                    //Initialises highest and lowest mark to first mark in the list

        int total = 0;
                                                                    //Adds up the marks
        int count = 0;
                                                                    //Counts the numbers of marks

        for (int[] row : marks) {
            for (int column : row) {
                total += column;
                count += 1;
                if (column > highestMark) {
                    highestMark = column;
                }
                else if (column < lowestMark) {
                    lowestMark = column;
                }
            }
        }
    }
}

```

```

    }
}

System.out.println("The highest mark is" + highestMark);
System.out.println("The lowest mark is" + lowestMark);
System.out.println("The average mark is" + total/count);
}
}

```

Activity 18

Here is the program implemented in Python:

```

gameScores = [['Alexis', 1, 19], ['Seema', 1, 29], ['Seema', 2, 44], ['Lois', 1, 10],
['Alexis', 2, 17], ['Alexis', 3, 36], ['Dion', 1, 23], ['Emma', 1, 27],
['Emma', 2, 48]]
highestL1Score = 0
highestL1Player = ''
highestL2Score = 0
highestL2Player = ''
highestL3Score = 0
highestL3Player = ''

for row in gameScores:
    player = row[0]
    level = row[1]
    score = row[2]

    if level == 1 and score > highestL1Score:
        highestL1Score = score
        highestL1Player = player
    elif level == 2 and score > highestL2Score:
        highestL2Score = score
        highestL2Player = player
    elif level == 3 and score > highestL3Score:
        highestL3Score = score
        highestL3Player = player

print('The highest score in Level 1 was', highestL1Score, 'achieved by',
highestL1Player)
print('The highest score in Level 2 was', highestL2Score, 'achieved by',
highestL2Player)
print('The highest score in Level 3 was', highestL3Score, 'achieved by',
highestL3Player)

```

Here is the algorithm implemented in C#:

```

string[,] marks = new string[,]
{
    {"Alexis", "1", "19"},

```

```

        {"Seema", "1", "29"},
        {"Seema", "2", "44"},
        {"Lois", "1", "10"},
        {"Alexis", "2", "17"},
        {"Alexis", "3", "36"},
        {"Dion", "1", "23"},
        {"Emma", "1", "27"}
    };

```

```

    int highestL1Score = 0, highestL2Score = 0, highestL3Score = 0;
    string highestL1Player = "", highestL2Player = "", highestL3Player = "";

```

// go through each row at a time and check if highest score for each level

```

    for (int row = 0; row <= 7; row++)
    {
        int level = 0, score = 0;
        string player = "";
        player = marks[row, 0];
        level = int.Parse(marks[row, 1]);
        score = int.Parse(marks[row, 2]);

        if (level == 1 && score > highestL1Score)
        {
            highestL1Score = score;
            highestL1Player = player;
        }
        else if (level == 2 && score > highestL2Score)
        {
            highestL2Score = score;
            highestL2Player = player;
        }
        else if (level == 3 && score > highestL3Score)
        {
            highestL3Score = score;
            highestL3Player = player;
        }
    }

```

```

    Console.WriteLine("The highest score in Level 1 was" + highestL1Score +
        "achieved by" + highestL1Player);

```

```

    Console.WriteLine("The highest score in Level 2 was" + highestL2Score +
        "achieved by" + highestL2Player);

```

```

    Console.WriteLine("The highest score in Level 3 was" + highestL3Score +
        "achieved by" + highestL3Player);

```


Here is the algorithm implemented in Java:

```
class Main {
    public static void main(String[] args) {
        // store all scores as a 2-D array of strings
        String[][] scores = new String[][] {
            {"Alexis", "1", "19"},
            {"Seema", "1", "29"},
            {"Seema", "2", "44"},
            {"Lois", "1", "10"},
            {"Alexis", "2", "17"},
            {"Alexis", "3", "36"},
            {"Dion", "1", "23"},
            {"Emma", "1", "27"},
            {"Emma", "2", "48"}
        };

        // store all high scores
        String[][] highScores = new String[][] {
            {"Level", "Player", "Highscore"},
            {"1", "Nobody", "0"},
            {"2", "Nobody", "0"},
            {"3", "Nobody", "0"},
        };

        // search through the 2-D array to find the player with the highest scores
        for(int i = 0; i < scores.length; i++) {
            // access player data from 2-D array
            String playerName = scores[i][0];
            int level = Integer.parseInt(scores[i][1]);
            int score = Integer.parseInt(scores[i][2]);

            // compare player data with current high scores
            int currentHighScore = Integer.parseInt(highScores[level][2]);
            if(score > currentHighScore) {
                highScores[level][1] = playerName;
                highScores[level][2] = String.valueOf(score);
            }
        }

        // display all high scores
        for(int row = 0; row < highScores.length; row++) {
            for(int col = 0; col < highScores[row].length; col++) {
                System.out.print(highScores[row][col] + ",");
            }

            // put each level on a new line
            System.out.println();
        }
    }
}
```

C#:

```

string[,] marks = new string[,]
{
    {"Alexis", "1", "19"},
    {"Seema", "1", "29"},
    {"Seema", "2", "44"},
    {"Lois", "1", "10"},
    {"Alexis", "2", "17"},
    {"Alexis", "3", "36"},
    {"Dion", "1", "23"},
    {"Emma", "1", "27"}
};

int highestL1Score = 0, highestL2Score = 0, highestL3Score = 0;
string highestL1Player = "", highestL2Player = "", highestL3Player = "";

// go through each row at a time and check if highest score for each level
for (int row = 0; row <= 7; row++)
{
    int level = 0, score = 0;
    string player = "";
    player = marks[row, 0];
    level = int.Parse(marks[row, 1]);
    score = int.Parse(marks[row, 2]);

    if (level == 1 && score > highestL1Score)
    {
        highestL1Score = score;
        highestL1Player = player;
    }
    else if (level == 2 && score > highestL2Score)
    {
        highestL2Score = score;
        highestL2Player = player;
    }
    else if (level == 3 && score > highestL3Score)
    {
        highestL3Score = score;
        highestL3Player = player;
    }
}

Console.WriteLine("The highest score in Level 1 was" + highestL1Score + "achieved by" + highestL1Player);
Console.WriteLine("The highest score in Level 2 was" + highestL2Score + "achieved by" + highestL2Player);
Console.WriteLine("The highest score in Level 3 was" + highestL3Score + "achieved by" + highestL3Player);

```

Java:

```
class Main {
    public static void main(String[] args) {
        // store marks in a 2-D array
        int[][] marks = {
            {89, 34, 59, 80},
            {64, 47, 11, 31},
            {91, 13, 56, 29},
            {0, 48, 61, 55},
            {91, 81, 78, 75}
        };

        // iterate through each set of marks
        for(int i = 0; i < marks.length; i++) {
            System.out.println("Marks" + i);

            // keep track as we loop through each value
            int highest = marks[i][0];
            int lowest = marks[i][0];
            int subtotal = 0;

            // loop through each score in a set of marks
            for(int j = 0; j < marks[i].length; j++) {

                // update highest score found so far
                if(marks[i][j] > highest) {
                    highest = marks[i][j];
                }

                // update lowest score found so far
                if(marks[i][j] < lowest) {
                    lowest = marks[i][j];
                }

                // update the subtotal so we can work out the average
                subtotal += marks[i][j];
            }

            // display stats for this set of marks
            System.out.println("Highest mark:" + highest);
            System.out.println("Lowest mark:" + lowest);
            System.out.println("Average mark:" + (subtotal / marks[i].length));
        }
    }
}
```

This will output the following:

Marks 0

Highest mark: 89

Lowest mark: 34

Average mark: 65

Marks 1

Highest mark: 64

Lowest mark: 11

Average mark: 38

Marks 2

Highest mark: 91

Lowest mark: 13

Average mark: 47

Marks 3

Highest mark: 61

Lowest mark: 0

Average mark: 41

Marks 4

Highest mark: 91

Lowest mark: 75

Average mark: 81

C#:

```
int[,] results = new int[,] { { 89, 34, 59, 80 }, { 64, 47, 11, 31 }, { 91, 13, 56, 29 },
{ 0, 48, 61, 55 }, { 91, 81, 78, 75 } };
```

```
int max, min, average, total, count;
```

```
max = 0;
```

```
min = 9999;
```

```
total = 0;
```

```
count = 0;
```

```
foreach (int currentValue in results)
```

```
{
```

```
    // find maximum value
```

```
    if (currentValue > max)
```

```
    {
```

```
        max = currentValue;
```

```
    }
```

```
    // find minimum value
```

```
    if (currentValue < min)
```

```
    {
```

```
        min = currentValue;
```

```
    }
```

```
    total += currentValue;
```

```
    count += 1;
```

```
}
```

```
average = total / count;
```

```
Console.WriteLine("Max:" + max + "Min:" + min + "Average:" + average);
```

ACTIVITY 19

- 1 **a** string
- b** string
- c** integer
- d** string

2 Here is the program implemented in Python:

```
albumCollection = [['Where Rivers Meet', 'Z Rahman', 2008, 'World'], ['Best of Cat
Stevens', 'C Stevens', 1984, 'Pop'], ['Come Away With Me', 'N Jones', 2012, 'Pop'],
['Shine', 'Bond', 2002, 'Instrumental'], ['Blessing', 'J Rutter', 2012, 'Classical']]
anotherGo = 'y'
while anotherGo == 'y':
    while True:
        choice = input("Press 'e' to enter details of a new album, or 's' to search
for an album.")
        if choice == 'e' or choice == 's':
            break

        if choice == 'e':
            newAlbum = []
            album = input('Enter the name of the album:')
            artist = input('Enter the name of the artist:')
            year = int(input('Enter the year of release:'))
            genre = input('Enter the genre:')
            newAlbum = [album, artist, year, genre]
            albumCollection.append(newAlbum)
        else:
            searchAlbum = input('Enter the title of the album:')
            found = False
            index = 0

            while found == False and index <= (len(albumCollection) - 1):
                if albumCollection[index][0] == searchAlbum:
                    print('Artist:', albumCollection[index][1], '\nYear of release:',
albumCollection[index][2])found = True
                else:
                    index = index + 1

            if found == False:
                print('This album is not in your collection.')

        anotherGo = input("Press 'y' if you want another go.")
```

Here is the algorithm implemented in C#:

```
using System;
```

```
namespace c_sharp_struct
```

```
{
    public class Program
    {
```

//structure definition must be within class definition rather than subroutine

```
        struct Album
        {
```

```

        public string title;
        public string artist;
        public int year;
        public string genre;
    }

```

```

    public static void Main()
    {
        char anotherGo = 'y';
        string choice;
        Album[] collection = new Album[5];

```

// add data to each structure in the array

// there are more efficient ways to do this but this is the simplest method

// students may wish to research constructors for a more efficient method

```

        collection[0].title = "Where Rivers Meet";
        collection[0].artist = "Z Rahman";
        collection[0].year = 2008;
        collection[0].genre = "World";

```

```

        collection[1].title = "Best of Cat Stevens";
        collection[1].artist = "C Stevens";
        collection[1].year = 1984;
        collection[1].genre = "Pop";

```

```

        collection[2].title = "Come Away with Me";
        collection[2].artist = "N Jones";
        collection[2].year = 2012;
        collection[2].genre = "Pop";

```

```

        collection[3].title = "Shine";
        collection[3].artist = "Bond";
        collection[3].year = 2002;
        collection[3].genre = "Instrumental";

```

```

        collection[4].title = "Blessing";
        collection[4].artist = "J Rutter";
        collection[4].year = 2012;
        collection[4].genre = "Classical";

```

```

        while (anotherGo == 'y')
        {

```

```

            while (true)
            {

```

```

                Console.WriteLine("Press e to enter details of a new album, or  
s to search for an album");

```

```

                choice = Console.ReadLine();

```

```

                if (choice == "e" || choice == "s")

```

```
{
    break;
}
}

    if (choice == "e")
    {
        int currentLength = collection.Length;
        int newLength = currentLength + 1; //Resize the array to allow new details to
                                           be added
        Array.Resize(ref collection, newLength);

        Console.WriteLine("Enter title:");
        collection[newLength - 1].title = Console.ReadLine();
                                           // -1 is to account for array starting at 0
        Console.WriteLine("Enter artist:");
        collection[newLength - 1].artist = Console.ReadLine();
        Console.WriteLine("Enter year:");
        string tempYearString = Console.ReadLine();
        collection[newLength - 1].year = int.Parse(tempYearString);
        Console.WriteLine("Enter genre:");
        collection[newLength - 1].genre = Console.ReadLine();
    }
    else
    {
        Console.WriteLine("Enter the title of the album:");
        string searchAlbum = Console.ReadLine();
        bool found = false;
        int index = 0;

        while (found == false && index < collection.Length)
        {
            if (collection[index].title == searchAlbum)
            {
                Console.WriteLine("Artist:" + collection[index].artist);
                Console.WriteLine("Year:" + collection[index].year);
                Console.WriteLine("Genre:" + collection[index].genre);
                found = true;
            }
            else
            {
                index++;
            }
        }

    }

    if (found == false)
    {
```

```
        Console.WriteLine("This album is not in your collection.");
    }
}
}
}
}
}
```

Here is the algorithm implemented in Java:

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        ArrayList<Object[]> albumCollection = new ArrayList<>();
        albumCollection.add(new Object[]{"Where Rivers Meet", "Z Rahman", 2008, "World"});
        albumCollection.add(new Object[]{"Best of Cat Stevens", "C Stevens", 1984, "Pop"});
        albumCollection.add(new Object[]{"Come Away With Me", "N Jones", 2012, "Pop"});
        albumCollection.add(new Object[]{"Shine", "Bond", 2002, "Instrumental"});
        albumCollection.add(new Object[]{"Blessing", "J Rutter", 2012, "Classical"});

        String choice;
        Scanner scan = new Scanner(System.in);
        String anotherGo = "y";

        while (anotherGo.equals("y")) {
            while (true) {
                System.out.println("Press 'e' to enter details of a new album, or 's' to
                search for an album.");
                choice = scan.nextLine();
                if (choice.equals("e") || choice.equals("s")) {
                    break;
                }
            }
            if (choice.equals("e")) {
                System.out.println("Enter the name of the album:");
                String album = scan.nextLine();
                System.out.println("Enter the name of the artist:");
                String artist = scan.nextLine();
                System.out.println("Enter the year of release:");
                int year = scan.nextInt();
                scan.nextLine(); //Necessary as nextInt() skips the new line
                System.out.println("Enter the genre:");
                String genre = scan.nextLine();
                Object[] newAlbum = {album, artist, year, genre};
                albumCollection.add(newAlbum);
            }
            anotherGo = scan.nextLine();
        }
    }
}
```



```

        } else {
            System.out.println("Enter the title of the album:");
            String searchAlbum = scan.nextLine();
            boolean found = false;
            int index = 0;

            while (found == false && index <= (albumCollection.size() - 1)) {
                if (albumCollection.get(index)[0].equals(searchAlbum)) {
                    Object[] album = albumCollection.get(index);
                    System.out.println("Artist:" + album[1] + "\nYear of release:" +
                        album[2]);
                    found = true;
                } else {
                    index = index + 1;
                }
            }

            if (found == false) {
                System.out.println("This album is not in your collection.");
            }
        }

        System.out.println("Press 'y' if you want another go.");
        anotherGo = scan.nextLine();
    }
    scan.close();
}
}

```

CHECKPOINT

S1 0

S2 There's a detailed explanation of how a linear search algorithm works in Unit 1, pages 19 and 20. And students implemented the algorithm themselves in Activity 16.

S3 Indexing in two-dimensional arrays is described in the worked example box on pages 66 and 67.

S4 Using a nested loop to traverse a two-dimensional array is illustrated on page 62.

C1 The address book is best represented by an array of arrays. Here is the program implemented in Python:

```

addressBook = [['Andrew Smith', 'as@buzz.com'], ['Maddison Jones', 'madJon@gmool.co.uk'],
['Bert Tintwhistle', 'BTWhistle@vgmail.com'], ['Richard Burton', 'RichBurt@ct.com'],
['Ann Mills', 'ann@mills.co.uk']]
anotherGo = 'y'

while anotherGo == 'y':
    searchName = input('Enter a name:')
    found = False
    index = 0

    while found == False and index <= (len(addressBook) - 1):
        if addressBook[index][0] == searchName:

```

```

        print('The email address of', searchName, 'is',
              addressBook[index][1])
        found = True
    else:
        index = index + 1

if found == False:
    print('This contact is not in your address book.')
anotherGo = input("\nPress 'y' if you want another go.")

```

Here is the algorithm implemented in C#:

```

string[,] addressBook = new string[,]
{
    {"Andrew Smith", "as@buzz.com"},
    {"Maddison Jone", "madjon@gmool.co.uk"},
    {"Bert Tintwhistle", "BTWhistle@vgmail.com"},
    {"Richard Burton", "RichBurt@ct.com"},
    {"Ann Mills", "ann@mills.co.uk"}
};

string anotherGo = "y", searchName = "";
bool found = false;
int index;

while (anotherGo.ToLower() == "y")
{
    index = 0;
    Console.WriteLine("Enter a name:");
    searchName = Console.ReadLine();
    found = false;

    while (found == false && index < addressBook.GetLength(0))
    {
        if (addressBook[index,0] == searchName)
        {
            Console.WriteLine("The email address of" + searchName
                              + "is" + addressBook[index,1]);
            found = true;
        }
        else
        {
            index += 1;
        }
    }

    if (found == false)
    {

```

```
        Console.WriteLine("This contact is not in your address book.");
```

```
    }
```

```
        Console.WriteLine("To search again enter y.");
```

```
        anotherGo = Console.ReadLine();
```

```
    }
```

Here is the algorithm implemented in Java:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String[][] addressBook = {{ "Andrew Smith", "as@buzz.com"}, {"Maddison Jones",
        "madJon@gmool.co.uk"}, {"Bert Tintwhistle", "BTWhistle@vgmail.com"}, {"Richard
        Burton", "RichBurt@ct.com"}, {"Ann Mills", "ann@mills.co.uk"}};
        String anotherGo = "y";

        Scanner scan = new Scanner(System.in);

        while (anotherGo.equals("y")) {
            System.out.println("Enter a name:");
            String searchName = scan.nextLine();
            boolean found = false;
            int index = 0;

            while (found == false && index <= (addressBook.length - 1)) {
                if (addressBook[index][0].equals(searchName)) {
                    System.out.println("The email address of" + searchName + "is" +
                    addressBook[index][1]);
                    found = true;
                } else {
                    index = index + 1;
                }
            }

            if (found == false) {
                System.out.println("This contact is not in your address book.");
            }

            System.out.println("Press 'y' if you want another go.");
            anotherGo = scan.nextLine();
        }

        scan.close();
    }
}
```

C2 There is no single right answer to this challenge. Here is one attempt implemented in Python:

```
import random
anotherGo = 'y'
```

```

while anotherGo == 'y' or anotherGo == 'Y':
    locRow = random.randint(0, 3)
    locCol = random.randint(0, 3)
    found = False
    print('\nThe treasure map consists of a grid of four rows and four columns.
Indexing begins at 0.\n')

while not found:
    guessRow = int( input('Enter a row:'))
    guessCol = int(input('Enter a column:'))
    print(guessRow, guessCol)
    if guessRow == locRow and guessCol == locCol:
        found = True
        print('Well done. You've found the treasure.')
    elif guessRow == locRow:
        print('Right row, wrong column. Try again.')
    elif guessCol == locCol:
        print('Right column, wrong row. Try again.')
    else:
        print('Nowhere near. Try again.')
anotherGo = input('\nDo you want another go?')

```

Here is the algorithm implemented in C#:

```

class Program
{
    static Random random = new Random();
    static void Main(string[] args)
    {
        int locRow = random.Next(0, 3);
        int locCol = random.Next(0, 3);
        int guessRow, guessCol;
        bool found = false;

        Console.WriteLine("The treasure map consists of a grid of four rows and four
columns.Indexing begins at 0.");

        Console.WriteLine(locRow + " " + locCol);

        while (!found) // while not found
        {
            Console.WriteLine("Enter a row:");
            guessRow = int.Parse(Console.ReadLine());

            Console.WriteLine("Enter a column:");
            guessCol = int.Parse(Console.ReadLine());

            if (guessRow == locRow && guessCol == locCol)

```

```

        {
            found = true;
            Console.WriteLine("Well done. You found the treasure!");
        }
        else if (guessRow == locRow)
        {
            Console.WriteLine("Correct row, wrong column. Try again.");
        }
        else if (guessCol == locCol)
        {
            Console.WriteLine("Correct column, wrong row. Try again.");
        }
        else
        {
            Console.WriteLine("Nowhere near. Try again.");
        }
    }
}
}
}

```

Here is the algorithm implemented in Java:

```

import java.util.*;

public class Main {
    public static void main(String[] args) {
        Random random = new Random();
        String anotherGo = "y";
        Scanner scan = new Scanner(System.in);

        while (anotherGo.equals("y") || anotherGo.equals("Y")) {
            int locRow = random.nextInt(3);
            int locCol = random.nextInt(3);
            boolean found = false;
            System.out.print("\nThe treasure map consists of a grid of four rows and four
            columns. Indexing begins at 0.\n");

            while (found == false) {
                System.out.println("Enter a row:");
                int guessRow = scan.nextInt();
                System.out.println("Enter a column:");
                int guessCol = scan.nextInt();
                System.out.println(guessRow + " " + guessCol);

                if (guessRow == locRow && guessCol == locCol) {
                    found = true;

```

```

        System.out.println("Well done. You've found the treasure.");
    }
    else if (guessRow == locRow) {
        System.out.println("Right row, wrong column. Try again.");
    }
    else if (guessCol == locCol) {
        System.out.println("Right column, wrong row. Try again.");
    }
    else {
        System.out.println("Nowhere near. Try again.");
    }
}

System.out.println("Do you want another go?");
anotherGo = scan.nextLine();
}

scan.close();
}
}

```

9 INPUT/OUTPUT

ACTIVITY 20

Here is the range check algorithm implemented in Python:

```

validNum = False
while validNum == False:
    number = int( input('Please enter a number between 1 and 10:'))
    if number >= 1 and number <= 10:
        validNum = True
        print('You have entered:', number)

```

Here is the range check algorithm implemented in C#:

```

bool validNum = false;
int number = 0;

while (validNum == false)
{
    Console.WriteLine("Enter a number between 1 and 10:");
    number = int.Parse(Console.ReadLine());

    if (number >= 1 && number <= 10)
    {
        validNum = true;
    }
}

Console.WriteLine("You have entered:" + number);

```

Here is the range check algorithm implemented in Java:

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        boolean validNum = false;
        int number = 0;

        while(validNum == false) {
            System.out.print("Please enter a number between 1 and 10:");
            number = scanner.nextInt();
            if(number >= 1 && number <= 10) {
                validNum = true;
            }
        }

        System.out.println("You have entered:" + number);
        scanner.close();
    }
}
```

ACTIVITY 21

Here is the presence check algorithm implemented in Python:

```
userName = ''
while userName == '':
    userName = input('Enter a username:')
    print('Hello', userName)
```

Here is the presence check algorithm implemented in C#:

```
string userName = "";

while (userName == "")
{
    Console.WriteLine("Enter a username:");
    userName = Console.ReadLine();
}

Console.WriteLine("Hello" + userName);
```

Here is the presence check algorithm implemented in Java:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String userName = "";
        Scanner scan = new Scanner(System.in);
```

```

        while (userName.equals("")) {
            System.out.print("Enter a username:");
            userName = scan.nextLine();
        }

        System.out.print("Hello" + userName);
        scan.close();
    }
}

```

ACTIVITY 22

Here is the look-up check algorithm implemented in Python. It uses the keyword 'in' to check if the form entered matches one of those stored in `arrayForms`. This avoids looping and so enables the code to be much shorter and simpler than the pseudocode example given in the question.

```

arrayForms = ['7AXB', '7PDB', '7ARL', '7JEH']
form = input('Enter a form:')
if form in arrayForms:
    print('Valid form.')
else:
    print('The form you entered does not exist.')

```

Here is the look-up check implemented in C#. It uses the `Array.Exists` method which allows the code to be simpler and shorter than the pseudocode example given in the question.

```

string form = "";
string[] arrayForms = {"7AXB", "7PDB", "7ARL", "7JEH"};

Console.WriteLine("Enter a form:");
form = Console.ReadLine();
if (Array.Exists(arrayForms, element => element == form))
{
    Console.WriteLine("Valid form.");
}
else
{
    Console.WriteLine("The form you entered does not exist.");
}

```

Here is the look-up check implemented in Java:

```

import java.util.*;
public class Main {
    public static void main(String[] args) {

        String[] arrayForms = {"7AXB", "7PDB", "7ARL", "7JEH"};

        System.out.print("Enter a form:");

        Scanner scan = new Scanner(System.in);
    }
}

```



```

String form = scan.nextLine();
scan.close();

if (Arrays.asList(arrayForms).contains(form)) {
    System.out.print("Valid form.");
}
else {
    System.out.print("The form you entered does not exist.");
}
}
}
}

```

ACTIVITY 23

Here is the length check algorithm implemented in Python:

```

postCode = input('Enter a postcode:')
length = len(postCode)
if length >= 6 and length <= 8:
    print('Valid')
else:
    print('Invalid')

```

Here is the length check algorithm implemented in C#:

```

string postCode = "";
int length = 0;

Console.WriteLine("Enter a postcode:");
postCode = Console.ReadLine();
length = postCode.Length;
if (length >= 6 && length <= 8)
{
    Console.WriteLine("Valid.");
}
else
{
    Console.WriteLine("Invalid.");
}

```

Here is the length check algorithm implemented in Java:

```

import java.util.*;
public class Main {
    public static void main(String[] args) {

        System.out.print("Enter a postcode:");
    }
}

```

```

    Scanner scan = new Scanner(System.in);
    String postCode = scan.nextLine();
    scan.close();

    int length = postCode.length();
    if(length >= 6 && length <= 8) {
        System.out.print("Valid");
    }
    else {
        System.out.print("Invalid");
    }
}
}

```

ACTIVITY 24

Python:

```

scores = [36, 39, 48, 54, 60, 69]

myFile = open('scores.txt', 'w')
for index in range(len(scores)):
    myFile.write(str((scores[index])) + ',')
myFile.close()

newScores = []
myFile = open('scores.txt', 'r')
newScores = myFile.read().split(',')
myFile.close()

print(newScores)

```

Java:

```

import java.util.*;
import java.io.*;
class Main {
    public static void main(String[] args) {
        int[] scores = {36, 39, 48, 54, 60, 69};

        // write values to file
        try {
            FileWriter myFile = new FileWriter("scores.txt");
            for(int index = 0; index < scores.length; index++) {
                myFile.write(scores[index] + ",");
            }
            myFile.close();
        } catch(IOException e) {
            System.out.println("Could not write to file");
        }
    }
}

```

```

// read values to file
try {
    File myFile = new File("scores.txt");
    Scanner scanner = new Scanner(myFile);
    String line = scanner.nextLine();
    scanner.close();
    String[] newScores = line.split(",");

    // display values that have been read from the file
    for(int i = 0; i < newScores.length; i++) {
        System.out.println(newScores[i]);
    }

} catch(FileNotFoundException e) {
    System.out.println("Could not read from file");
}
}
}

```

C#:

// Writing to a file

// remember to add "using System.IO"

```

int[] grades = new int[] {36, 39, 48, 54, 60, 69 };
StreamWriter sw = new StreamWriter("grades.csv");
for (int i = 0; i < grades.GetLength(0); i++)
{
    sw.WriteLine(grades[i] + ",");
}

```

//may need to add drive and path here

```
sw.Close();
```

// read all of file into array

// remember to add "using System.IO"

```

string[] allLines = System.IO.File.ReadAllLines("grades.csv"); //may need to add drive and path here
int length = allLines.Length;
int[] grades;
grades = new int[length];

// for each line in array containing all lines from the file, split using delimiter and assign to array
for (int i = 0; i < length; i++)
{
    string[] temp = allLines[i].Split(',');
    grades[i] = Int32.Parse(temp[0]);
}

```

ACTIVITY 25

Python:

```

scores = [['Faruq',36], ['Jalila', 39], ['Isam', 48], ['Inaya', 54], ['Kadim',
60], ['Jumana', 69]]

```

```

myFile = open('scores.txt', 'w')
for index in range(len(scores)):
    myFile.write(scores[index][0] + ',')
    new = str(scores[index][1])
    myFile.write(new + ',')
myFile.close()

newScores = []
myFile = open('scores.txt', 'r')
newScores = myFile.read().split(',')
myFile.close()

newScores_2 = []
for index in range(0, len(newScores)-1, 2):
    newScores_2.append([newScores[index], newScores[index+1]])

print(newScores_2)

```

Java:

```

import java.util.*;
import java.io.*;
class Main {
    public static void main(String[] args) {
        String[][] scores = {
            {"Faruq", "36"},
            {"Jalila", "39"},
            {"Isam", "54"},
            {"Inaya", "60"},
            {"Jumana", "69"}};

        // write values to file
        try {
            FileWriter myFile = new FileWriter("scores.txt");
            for(int index = 0; index < scores.length; index++) {

                // write name
                myFile.write(scores[index][0] + ",");

                // write score
                myFile.write(scores[index][1] + "\n");
            }
            myFile.close();
        } catch(IOException e) {
            System.out.println("Could not write to file");
        }

        // read values to file
        try {
            ArrayList <String[]> newScores = new ArrayList<String[]>();
            File myFile = new File("scores.txt");
            Scanner scanner = new Scanner(myFile);
            while(scanner.hasNextLine()) {

```

```

        String line = scanner.nextLine();
        String[] parts = line.split(",");
        newScores.add(parts);
    }
    scanner.close();

    // display values that have been read from the file
    for(int i = 0; i < newScores.size(); i++) {
        System.out.println(newScores.get(i)[0] + ":" + newScores.get(i)[1]);
    }

    } catch(FileNotFoundException e) {
        System.out.println("Could not read from file");
    }
}
}

```

C#:

```

string[,] grades = new string[,] { { "Faruq", "60"}, { "Jalila", "90"} };

StreamWriter sw = new StreamWriter("grades.csv");

for (int i = 0; i < grades.GetLength(0); i++)
{
    sw.WriteLine(grades[i, 0] + "," + grades[i,1]);
}

sw.Close();

```

Reading back in to 2-D array:

```

// read all of file into array
string[] allLines = System.IO.File.ReadAllLines("grades.csv");
int length = allLines.Length;

string[,] grades;
grades = new string[length, 2];

// for each line in array containing all lines from the file, split using delimiter and assign to 2-D array
for (int i = 0; i < length; i++)
{
    string[] temp = allLines[i].Split(',');
    grades[i, 0] = temp[0];
    grades[i, 1] = temp[1];
}

```

CHECKPOINT

- S1** The 'garbage in, garbage out' principle is explained on page 70.
- S2** Gordon Weidmann, 30 May 1954, purple.
- S3** The advantage of writing data to a text file is explained on page 73.

C1 Here's the program implemented in Python. It's worth revisiting it once students have studied the next section of this chapter and getting them to redesign it using subprograms. The data file 'employees.txt' is supplied with this pdf file.

```
departments = ['Sales', 'Production', 'Design', 'Finance', 'Cust Service']
```

```
#Menu
```

```
print("\n_____MENU_____\n")
```

```
print("E: Enter an employee's details\n")
```

```
print("S: Search for an employee's details\n")
```

```
print("Enter 'Q' to quit the program\n")
```

```
validChoice = True
```

```
while validChoice:
```

```
    optionsChoice = input('\nPlease enter an option (E, S or Q):')
```

```
    if optionsChoice == "e" or optionsChoice == "E":
```

```
        employees = open("employees.txt", "a")    #Uses append tag ("a") to avoid overwriting
                                                    existing records
```

```
        while True:
```

```
            employeeNumb = input("\nEnter employee's number:")
```

```
            if len(employeeNumb) == 3:            #Checks that employeeNumb is three digits long
```

```
                break
```

```
            employeeName = input("\nEnter employee's name:")
```

```
            while True:
```

```
                employeeDept = input("\nEnter employee's department:")
```

```
                if employeeDept in departments:
```

```
                    break
```

```
            newRecord = employeeNumb & "," & employeeName & "," & employeeDept &
```

```
            "\n"
```

```
            employees.write(newRecord)
```

```
            employees.close()
```

```
elif optionsChoice == "s" or optionsChoice == "S":
```

```
    rawData = open("employees.txt", "r")
```

```
    inputData = rawData.readlines()
```

```
    rawData.close()
```

#Splits each line of rawData into a list of strings and appends each list to the 2-dimensional array `employeeRecords`

```
    employeeRecords = []
```

```
    index = 0
```

```
for line in inputData:
```

```
    employeeRecords.append(inputData[index].split(","))
```

```
    index += 1
```

```
#Linear search used to find employee in file
```

```
searchID = input("\nEnter the employee's number:")
```

```
found = False
```

```
index = 0
```

```

while found == False and index <= len(employeeRecords):
    if employeeRecords[index][0] == searchID:
        print('\nThis is', employeeRecords[index][1], 'who works in',
              employeeRecords[index][2])
        found = True
    else:
        index += 1

    if found == False:
        print('This employee is not in the file.')
    elif optionsChoice == "q" or optionsChoice == "Q":
        print ('\nYou have opted to quit the program.' )
        validChoice = False
    else:
print('Invalid selection.')

```

Here is the program implemented in C#:

```

string fileName = "employees.csv";
string[] departments = { "Sales", "Production", "Design", "Finance", "Cust
Service" };
string employeeDept, employeeName, employeeNumb, optionsChoice;
bool validChoice = true;

//Menu
Console.WriteLine("_____MENU_____");
Console.WriteLine("E: Enter an employee's details");
Console.WriteLine("S: Search for an employee's details");
Console.WriteLine("Enter Q to quit the program");

while (validChoice)
{
    Console.WriteLine("Please enter an option (E, S or Q):");
    optionsChoice = Console.ReadLine();

    if (optionsChoice.ToLower() == "e")
    {
        // get employee details, validate and append to file
        while (true)
        {
            Console.WriteLine("Enter employee's number:");
            employeeNumb = Console.ReadLine();
            if (employeeNumb.Length == 3) //checks three digits have been entered
            {
                break;
            }
        }
    }
}

```

```

        Console.WriteLine("Enter employee's name:");
        employeeName = Console.ReadLine();

```

```

        while (true)
        {

```

```

            Console.WriteLine("Enter employee's department:");
            employeeDept = Console.ReadLine();

```

```

            if (Array.Exists(departments, element => element == employeeDept))
            {

```

```

                break;
            }
        }

```

```

        using (StreamWriter sw = File.AppendText(fileName)) //StreamWriter needs "Using
                                                                System.io"
        {

```

```

            sw.WriteLine(employeeNumb + "," + employeeName + "," + employeeDept);
        }
    }

```

```

    else if (optionsChoice.ToLower() == "s")
    {

```

```

        //open file and search for employee

```

```

        string[] lines = System.IO.File.ReadAllLines(fileName);
        int length = lines.Length;

```

```

        //Array is declared as having the correct number of rows from length of file and 3 columns

```

```

        string[,] employeeRecords = new string[length, 3];

```

```

        for (int i = 0; i <= (length - 1); i++)
        {

```

```

            string[] tempLine = lines[i].Split(',');

```

```

            employeeRecords[i, 0] = tempLine[0];

```

```

            employeeRecords[i, 1] = tempLine[1];

```

```

            employeeRecords[i, 2] = tempLine[2];
        }
    }

```

```

        Console.WriteLine("Enter the employee's number:");

```

```

        string searchID = Console.ReadLine();

```

```

        //Linear search to find employee in file

```

```

        int index = 0;

```

```

        bool found = false;

```

```

        while (found == false && index < length)
        {

```

```

            if (employeeRecords[index, 0] == searchID)
            {

```

```

                Console.WriteLine("This is" + employeeRecords[index, 1] + "who
                works in" + employeeRecords[index, 2]);

```

```

                found = true;
            }
        }
    }

```



```

    }
    else
    {
        index++;
    }
}
if (found == false)
{
    Console.WriteLine("The employee is not on file.");
}
}
else if (optionsChoice.ToLower() == "q")
{
    Console.WriteLine("You have opted to quit the program.");
    validChoice = false;
}
else
{
    Console.WriteLine("Invalid choice.");
}
}
}

```

Here is the program implemented in Java:

```

import java.io.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.*;
public class Main {
    public static void main(String[] args) {

        String[] departments = {"Sales", "Production", "Design", "Finance", "Cust Service"};

        //Menu
        System.out.print("\n_____MENU_____\n");
        System.out.print("E: Enter an employee's details\n");
        System.out.print("S: Search for an employee's details\n");
        System.out.print("Enter 'Q' to quit the program\n");
        boolean validChoice = true;
        Scanner scan = new Scanner(System.in);

        while (validChoice) {
            System.out.print("\nPlease enter an option (E, S or Q):");
            String optionsChoice = scan.nextLine();

            if (optionsChoice.equals("e") || optionsChoice.equals("E")) {
                try {
                    FileWriter employees = new FileWriter("employees.txt", true);

```

//The second argument is set to 'true' to enable append mode to avoid overwriting existing records

```

        String employeeNumb;
        while (true) {
            System.out.print("\nEnter employee's number:");
            employeeNumb = scan.nextLine();
            if (employeeNumb.length() == 3) {

                //Checks that employeeNumb is three digits long
                break;
            }
        }

        System.out.print("\nEnter employee's name:");
        String employeeName = scan.nextLine();

        String employeeDept;
        while (true) {
            System.out.print("\nEnter employee's department:");
            employeeDept = scan.nextLine();
            if (Arrays.asList(departments).contains(employeeDept)) {
                break;
            }
        }

        String newRecord = employeeNumb + "," + employeeName + "," +
            employeeDept + "\n";
        employees.write(newRecord);
        employees.close();

    } catch (IOException e) {
        e.printStackTrace();
    }
} else if (optionsChoice.equals("s") || optionsChoice.equals("S")) {
    List<String> inputData;
    try {
        inputData = Files.readAllLines(Paths.get("employees.txt"),
            StandardCharsets.UTF_8);

        //Splits each line of rawData into a list of strings and appends each list to the 2 dimensional
        array employeeRecords
        ArrayList<String[]> employeeRecords = new ArrayList<String[]>();
        int index = 0;

        for (String line : inputData) {
            employeeRecords.add(inputData.get(index).split(","));
            index += 1;
        }

        //Linear search used to find employee in file

```

```

        System.out.print("\nEnter the employee's number:");
        String searchID = scan.nextLine();
        boolean found = false;
        index = 0;

        while (found == false && index < employeeRecords.size()) {
            if (employeeRecords.get(index)[0].equals(searchID)) {
                System.out.print("\nThis is" + employeeRecords.get(index)[1]
                    + "who works in" + employeeRecords.get(index)[2]);
                found = true;
            } else {
                index += 1;
            }
        }

        if (found == false) {
            System.out.print("This employee is not in the file.");
        }

        } catch (IOException e) {
            e.printStackTrace();
        }
    } else if (optionsChoice.equals("q") || optionsChoice.equals("Q")) {
        System.out.print("\nYou have opted to quit the program.");
        validChoice = false;
    } else {
        System.out.print("Invalid selection.");
    }
}
scan.close();
}
}

```

10 SUBPROGRAMS

ACTIVITY 26

- a** rectangleLength, rectangleLength, rectangleArea
- b** area
- c** rectangleLength, rectangleWidth
- d** length, width

ACTIVITY 27

- a** The program provides a menu system so that users can log in, create a login, change a password. Each selection calls a subprogram.

b**Python:**

```

print ("1. Register as a new user.")
print ("2. Login.")
print ("3.Change your password.")
print ("4. Exit.")

correct = 0
while correct == 0:
    choice = int(input('Please select a menu option. '))
    if choice == 1:
        newUser()
        correct = 1
    elif choice == 2:
        login()
        correct = 1
    elif choice == 3:
        changePassword()
        correct = 1
    elif choice == 4:
        exit()
        correct = 1
    else:
        print('Incorrect option. Try again.')
```

Java:

```

import java.util.*;
class Main {

    // procedures can be implemented later
    // just shown here to illustrate the structure

    public static void newUser() {}
    public static void login() {}
    public static void changePassword() {}
    public static void exit() {}
    public static void main(String[] args) {
        boolean running = true;
        Scanner scanner = new Scanner(System.in);

        while(running) {

            // display main menu
            System.out.println("1. Register as a new user");
            System.out.println("2. Login");
            System.out.println("3. Change your password");
            System.out.println("4. Exit");

            // get user input without crashing if they don't enter an integer
            int choice = 0;
```

```

        if(scanner.hasNextInt()) {
            choice = scanner.nextInt();
        } else {
            scanner.nextLine();
        }

        // switch is the same as lots of IF/ELSE statements
        switch(choice) {

            // Register as a new user
            case 1:
                newUser();
            break;

            // login
            case 2:
                login();
            break;

            // change password
            case 3:
                changePassword();
            break;

            // exit
            case 4:
                running = false;
            break;

            // anything else
            default:
                System.out.println("Incorrect option, try again");
            break;
        }
        scanner.close();
    }
}

```

C#:

```

string choiceString;
int choice;
bool validChoice = false;

while (validChoice == false)
{
    Console.WriteLine("1. Register as a new user");
    Console.WriteLine("2. Login");
    Console.WriteLine("3. Change your password");
    Console.WriteLine("4. Exit");
}

```

```

    Console.WriteLine("Please select a menu option");
    choiceString = Console.ReadLine();
    choice = int.Parse(choiceString);

    if (choice == 1)
    {
        validChoice = true;
        newUser();
    }
    else if (choice == 2)
    {
        validChoice = true;
        login();
    }
    else if (choice == 3)
    {
        validChoice = true;
        changePassword();
    }
    else if (choice == 4)
    {
        validChoice = true;
        exit();
    }
    else
    {
        Console.WriteLine("Incorrect option. Try again");
    }
}

```

CHECKPOINT

- S1** The benefits of using subprograms are explained on page 90.
- S2** Variable scope is explained on page 83. Students could use Activity 27 to illustrate the scope of a variable.
- S3** If a local variable and a global variable share the same name, the subprogram will use the local variable and the global variable will not be changed.
- S4** There are loads to choose from. The two specified in Pearson Edexcel pseudocode are `LENGTH()` and `RANDOM()`. Others students may have come across in Python are `max()`, `int()`, `print()`, `range()`, `type()`, `str()`.
- C1** Here is the program implemented in Python. It makes use of the imported statistics module, which includes mean, median and mode methods. Error handling is incorporated to prevent the user from entering any non-integers and to stop the program from crashing if the mode can't be calculated for the given set of numbers.

```

import statistics
def enterSet():
    numbers = []
    anotherNumber = 'y'

```

```

while anotherNumber == 'y' or anotherNumber == 'Y':
    while True:
        nextNumber = int(input('\nNumber?'))
        numbers.append(nextNumber)
        break
    except ValueError:
        print('Your entry must be a valid integer. Please try again.')

    anotherNumber = input('\nAny more numbers to be entered?')

numbers.sort()
print('\nYou have entered this set of numbers', numbers)
return(numbers)

def menu():
    print("\n_____MENU_____")
    print("A: Mean\n")
    print("B: Median\n")
    print("C: Mode\n")
    print("Enter 'Q' to quit the program\n")
    validChoice = True

while validChoice:
    optionsChoice = input('\nPlease enter an option (A, B, C or Q):')
    if optionsChoice == "a" or optionsChoice == "A":
        print ('\nThe mean is', statistics.mean(numberSet))
    elif optionsChoice == "b" or optionsChoice == "B":
        print ('\nThe median is', statistics.median(numberSet))
    elif optionsChoice == "c" or optionsChoice == "C":
        while True:
            #Prevents program from crashing if mode can't be calculated for given set of numbers

            print ('\nThe mode is', statistics.mode(numberSet))
            break
        except ValueError:
            print('Not possible to calculate mode for this set of numbers.')
            break

    elif optionsChoice == "q" or optionsChoice == "Q":
        print ('\nYou have opted to quit the program.' )
        validChoice = False
    else:
        print('Invalid selection.')

```

#Main program

```
numberSet = enterSet()
    menu()
```

Here is the program implemented in C#:

```
static void Main(string[] args)
{

    bool isNumeric = false;
    int qtyInput = 0;

    // get quantity of values from user. Validated using TryParse.
    // TryParse returns a boolean value depending on validity of data for specified type
    while (!isNumeric)
    {
        Console.WriteLine("Enter the quantity of numbers you wish to enter:");
        isNumeric = int.TryParse(Console.ReadLine(), out qtyInput);
    }

    // declare array of correct size
    int[] numbers = new int[qtyInput];

    //get data from user, validated using TryParse
    for (int i = 0; i < qtyInput; i++)
    {
        isNumeric = false;
        while (!isNumeric)
        {
            Console.WriteLine("Enter value" + i);
            isNumeric = int.TryParse(Console.ReadLine(), out numbers[i]);
        }
    }

    //Menu
    Console.WriteLine("_____MENU_____");
    Console.WriteLine("A: Mean");
    Console.WriteLine("B: Median");
    Console.WriteLine("C: Mode");
    Console.WriteLine("Q: Quit");

    bool validChoice = true;
    string choice = "";
```



```
while (validChoice)
{
    Console.WriteLine("Enter A, B, C or Q");
    choice = Console.ReadLine();

    if (choice.ToLower() == "a")
    {
        Console.WriteLine("The mean is:" + Mean(numbers));
    }
    else if (choice.ToLower() == "b")
    {
        Console.WriteLine("The median is:" + Median(numbers));
    }
    else if (choice.ToLower() == "c")
    {
        Console.WriteLine("The mode is:" + Mode(numbers));
    }
    else if (choice.ToLower() == "q")
    {
        Console.WriteLine("Bye");
        validChoice = false;
    }
    else
    {
        Console.WriteLine("Invalid choice");
    }
}

static double Mean (int[] arr)
{
    return arr.Average(); // uses LINQ – Average in LINQ is the mean
}

static double Median (int[] arr)
{
    int numElements = arr.Length;
    double calcMedian = 0;
    bool isEven = numElements % 2 == 0;
    Array.Sort(arr);

    if (isEven)
    {
```

```

        calcMedian = (arr[numElements / 2 - 1] + arr[numElements / 2]) / 2.0d;
    }
    else
    {
        calcMedian = arr[(numElements / 2)];
    }

    return calcMedian;
}

static int Mode (int[] arr)
{
    // no built-in function for calculating mode in C#. This uses LINQ.
    int calcMode = arr.GroupBy(i => i)
        .OrderByDescending(g => g.Count())
        .Select(g => g.Key)
        .FirstOrDefault();

    return calcMode;
}

```

Here is the program implemented in Java:

```

import java.util.*;

public class Main {

    static ArrayList<Integer> numberSet;
    static Scanner scan;

    static Integer sum() {
        int total = 0;
        for (Integer number : numberSet) {
            total += number;
        }
        return total;
    }

    static float mean() {
        float mean = (float)sum() / numberSet.size();
        return mean;
    }

    static float median() {
        int middleIndex = numberSet.size()/2;

```

```

float median;
if(numberSet.size() % 2 == 0) {
    median = (numberSet.get(middleIndex) + numberSet.get(middleIndex-1))/2.f;
} else {
    median = numberSet.get(middleIndex);
}
return median;
}

```

```

static Integer mode() {
    Integer mode = null;
    int maxCount = 0;
    int count = 0;
    int previous = numberSet.get(0);
    boolean modeApplicable = true;
    for (int number : numberSet) {
        if (previous == number) {
            count += 1;
            if (count > maxCount) {
                maxCount = count;
                mode = number;
                modeApplicable = true;
            }
        } else if(count == maxCount) {
            modeApplicable = false;
        }
    } else {
        count = 0;
    }
    previous = number;
}

if (modeApplicable && (maxCount != 1 || numberSet.size() == 1)) {
    return mode;
} else {
    return null;
}
}

```

```

static ArrayList<Integer> enterSet() {
    ArrayList<Integer> numbers = new ArrayList<Integer>();
    String anotherNumber = "y";

    while (anotherNumber.equals("y") || anotherNumber.equals("Y")) {
        while (true) {

```

//Prevent the user from entering non-integer values

```

        try {
            System.out.print("\nNumber?");
            int nextNumber = scan.nextInt();
            numbers.add(nextNumber);
            break;
        } catch (ArithmeticException e) {
            System.out.print("Your entry must be a valid integer. Please try
            again.");
        }
    }

    System.out.print("\nAny more numbers to be entered?");
    anotherNumber = scan.next();
}

Collections.sort(numbers);
System.out.print("\nYou have entered this set of numbers" + numbers);
return numbers;
}

static void menu() {
    System.out.print("\n_____MENU_____");
    System.out.print("\nA: Mean\n");
    System.out.print("\nB: Median\n");
    System.out.print("\nC: Mode\n");
    System.out.print("\nEnter 'Q' to quit the program\n");
    boolean validChoice = true;

    while (validChoice) {
        System.out.print("\nPlease enter an option (A, B, C or Q):");
        String optionsChoice = scan.next();

        if (optionsChoice.equals("a") || optionsChoice.equals("A")) {
            System.out.print("\nThe mean is" + mean());
        } else if (optionsChoice.equals("b") || optionsChoice.equals("B")) {
            System.out.print("\nThe median is" + median());
        } else if (optionsChoice.equals("c") || optionsChoice.equals("C")) {
            System.out.print("\nThe mode is" + mode());
        } else if (optionsChoice.equals("q") || optionsChoice.equals("Q")) {
            System.out.print("\nYou have opted to quit the program.");
            validChoice = false;
        } else {
            System.out.print("Invalid selection.");
        }
    }
}

```

```
//Main program
public static void main(String[] args) {
    scan = new Scanner(System.in);
    numberSet = enterSet();
    menu();
    scan.close();
}
}
```

11 TESTING AND EVALUATION

ACTIVITY 28

The selection is incorrect.

Any year group greater than 7 (e.g. 30) would be acceptable, as would any year group less than 13.

It should be `IS yearGroup >= 7 AND <= 13`.

ACTIVITY 29

Example 1:

```
SET index TO 1
WHILE index < 10 DO
    SEND index TO DISPLAY
    SET index TO index + 1
END WHILE
```

#Index must be incremented in the loop so that the terminating condition is eventually met

Example 2:

```
SET index TO 1
WHILE index < 10 DO
    SEND index TO DISPLAY
    SET index TO index + 1
END WHILE
```

#The variable index must increase each time through the loop, not decrease

Example 3:

```
SET index TO -5 #or any number less than 1
WHILE index < 1 DO
    SEND INDEX TO DISPLAY
    SET index TO index + 1
END WHILE
```

#The value of index must be less than 1 for the loop to execute

ACTIVITY 30

INDEX	NUMBER 1	NUMBER 2
	2	3
1	2	5
2	4	9
3	12	21
4	48	69
5	240	309

ACTIVITY 31

1

LENGTH	COUNT	INDEX	GENDER [INDEX]
10			
	0		
		0	
			M
		1	
			M
		2	
			F
	1		
		3	
			M
		4	
			F
	2		
		5	
			F
	3		
		6	
			M
		7	
			F
	4		
		8	
			M
		9	
			F
	5		
		10	

2

TOTAL	NUMBER	OUTPUT
0	3	
3	13	
16	21	
37	28	
65	0	65

ACTIVITY 32

The **RANDOM** function in Pearson Edexcel's pseudocode has only one parameter 'n' and generates numbers between 0 and 'n'.

```

1 SET number TO RANDOM(5) + 1
  SET guessed TO False
  WHILE guessed = False DO
    REPEAT
      SEND 'Enter a number between 1 and 6' TO DISPLAY
    UNTIL guess > 0 AND guess < 7 THEN
      IF guess = number THEN
        SEND 'Well Done.' TO DISPLAY
        SET guessed TO True
      ELSE
        SEND 'Try again.' TO DISPLAY
      END IF
    END WHILE
  END WHILE

```

2 The plan needs to test that:

- the program generates numbers in the correct range (1–6)
- user input is restricted to numbers between 1 and 6
- appropriate messages are output for correct and incorrect guesses
- the program terminates when a correct guess is entered and continues until that point.

3 Here is the algorithm implemented in Python:

```

import random
number = random.randint(1, 6)
guessed = False
WHILE guessed == False:
    WHILE True:
        guess = int(input('Enter a number between 1 and 6:'))
        IF guess > 0 and guess < 7:
            break

    IF guess == number:
        print('Well Done')
        guessed = True

```

ELSE:

```
print('Try again')
```

Here is the algorithm implemented in C#:

```
int guess = 0;
    string guessString;
    bool guessed = false;

    Random rnd = new Random();
    int number = rnd.Next(1, 7);

    while (guessed == false)
    {
        while (true)
        {
            Console.WriteLine("Guess a number between 1 and 6:");
            guessString = Console.ReadLine();
            guess = int.Parse(guessString);
            if (guess > 0 && guess < 7)
            {
                break;
            }
        }
    }

    if (guess == number)
    {
        Console.WriteLine("Well done");
        guessed = true;
    }
    else
    {
        Console.WriteLine("Try again.");
    }
```

Here is the algorithm implemented in Java:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Random random = new Random();
        Scanner scan = new Scanner(System.in);
        int number = random.nextInt(6) + 1;
        boolean guessed = false;
```



```

    while (guessed == false) {
        int guess;

        while (true) {
            System.out.print("Enter a number between 1 and 6:");
            guess = scan.nextInt();
            if (guess > 0 && guess < 7) {
                break;
            }
        }
        if (guess == number) {
            System.out.print("Well Done");
            guessed = true;
        } else {
            System.out.print("Try again");
        }
    }
    scan.close();
}
}

```

4 Solution not required.

CHECKPOINT

- S1** The three types of error are summarised in Table 2.14 on page 100.
S2 Trace tables were introduced on pages 21–22 of Unit 1 and are revisited on pages 94–96.
S3 Some features of IDEs are described on page 101, but students should provide examples from the IDE they use.
S4 The purpose of a test plan and how to design and implement a test plan are covered on pages 101–102.
S5 Normal, boundary and erroneous data are explained on page 100.
S6 Whenever an improvement is made to a program, there's a possibility that fresh errors are unwittingly introduced at the same time – hence the need to re-test.

C1–3 Students' own answers.

UNIT QUESTIONS

1 a	LINE NUMBER(S)
a. variable initialisation	01, 06, 07
b. type declaration	02
c. selection	09
d. iteration	08–12
e. data structure	01
f. subprogram	04–14

- b** The call to the subprogram `maxCalc` is on line 16.

2 The algorithm counts the number of marks of 50 or above in the array scores.

3

LENGTH	COUNT	INDEX	SCORES [INDEX]
5			
	0		
		0	
			45
		1	
			67
	1		
		2	
			34
		3	
			98
	2		
		4	
			52
	3		

4 a rectangle

b print

c length or width

d length, width, area or perimeter

e ar or per

f 8

5 a array

b If the entry and the brand names in the array are not in the same case, when the brands are being searched for they will not be found because comparisons are case sensitive.

c Please see file Q2b for solution.

- Loop set up to search through the list.
- Range of loop should be correct using length function.
- Should reference the correct index of the two-dimensional array.
- Brand names in list set to upper case.
- Variable used to signify if search has been successful.
- Position correctly worked out.
- Correct display of brand name and its position to user.
- Message informing the user if it is not present.

6 Marks for this question use the scheme as shown in the SAM.

UNIT 3: DATA

12 BINARY

ACTIVITY 1

- a $00111001 = 57$
- b $11000110 = 198$
- c $10101010 = 170$

ACTIVITY 2

- a $69 = 01000101$
- b $193 = 11000001$
- a $239 = 11101111$

ACTIVITY 3

- a $10011010 + 11010111 = 101110001$
- b $00001101 + 10101010 = 10110111$
- c $11010111 + 10001010 = 101100001$

ACTIVITY 4

- 1
 - a $-113 = 10001111$
 - b $-56 = 11001000$
 - c $-90 = 10100110$
- 2 $90 + -33 = 01011010 + 11011111 = 00111001$

ACTIVITY 5

- a $00111010 * 2^3 = 111010000$
- b $10011101 / 2^4 = 00001001$

ACTIVITY 6

1 & 2

- a $10010000 / 2^2 = 11100100$ ($-112 / 4 = -28$)
- b $11110110 / 2^1 = 11111011$ ($-10 / 2 = -5$)
- c $11000000 / 2^3 = 11111000$ ($-64 / 8 = -8$)

ACTIVITY 7

- 1
 - a $01101110 = 6E$
 - b $10011100 = 9C$
 - c $00101010 = 2A$
- 2
 - a $A6 = 10100110$
 - b $9C = 10011100$
 - c $2D = 00101101$

CHECKPOINT

S1 $11011001 + 10010010 = 101101011$ ($217 + 146 = 363$)

S2 $00011001 + 11110011 = 00001100$ ($25 + (-13) = 12$)

S3 1001001000 ($73 \times 8 = 584$)

S4 $11001101 = 205$ in denary and CD in hex.

C1 The use of binary to represent data and program instructions is explained on pages 110–111.

C2 The difference between a logical shift and an arithmetic shift is explained on pages 120–122.

13 DATA REPRESENTATION**ACTIVITY 8**

The ASCII code for 'ASCII code' is: 01000001 01010011 01000011 01001001 01001001 00100000 01100011 01101111 01100100 01100101 00101110

ACTIVITY 9

```
FUNCTION chr(number)
BEGIN FUNCTION
    SET arrayNumb2Ascii TO [[32, ''], [33, '!'], [34, '"'], [35, '#'], [36, '$'], [37,
    '%'], [38, '&'], .....[125, '']]
    #A 2-dimensional array containing all of
    #the denary ASCII codes and characters

    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
```

ACTIVITY 10

This algorithm is revisited later in the unit in the section 'Encryption'. You may want to postpone doing this activity until students have learned more about the Caesar cipher algorithm. Here's the program written in Python:

```
message = input('Enter the message to encrypt:')
shift = int(input('\nEnter the size of the shift:'))
secretMessage = ''

for character in message:
    number = ord(character)

    if character.lower() in 'abcdefghijklmnopqrstuvwxyz': #Checks that the character is a letter
                                                         #rather than a space or punctuation
                                                         #mark
        number += shift
        #deals with letters at start and end of
        #alphabet

    if character.isupper(): #It's upper case
        if number > ord('Z'):
            number -= 26
        elif number < ord('A'):
```

```

        number += 26
    else:
        if number > ord('z'):
            number -= 26
        elif number < ord('a'):
            number += 26
        secretMessage = secretMessage & chr(number)
else:
    secretMessage = secretMessage & character
    print (secretMessage)

```

#Must be lower case

#Non-letters are added unchanged

Here's the program written in C#:

//This implementation skips full stops and spaces. It also maintains case of letters.

```

string plaintext = "The ASCII code represents characters.";
string ciphertext = "";
for (int i = 0; i < plaintext.Length; i++)
{
    //don't encode full stops or spaces.
    if (plaintext[i] == '.' || plaintext[i] == ' ')
    {
        ciphertext += plaintext[i];
    }
    else
    {
        ciphertext += (char)((((plaintext[i] + 3 - plaintext[i]) % 26) + plaintext[i]));
    }
}
Console.WriteLine(ciphertext);

```

Here's the program written in Java:

```

import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter the message to encrypt:");
        String message = scan.nextLine();
        System.out.print("Enter the size of the shift:");
        int shift = scan.nextInt();
        String secretMessage = "";

        // loop through each character in the message
        for(int i = 0; i < message.length(); i++) {
            // convert to lowercase one letter at a time
            char character = Character.toLowerCase(message.charAt(i));

            // get ASCII value of character
            int number = character;

```

```

// only shift letters
if(Character.isLetter(character)) {
    number += shift;

    // wrap around the alphabet if necessary
    if(number > (int)'z') {
        number -= 26;
    }
    if(number < (int)'a') {
        number += 26;
    }
}

// convert ASCII number back to character and add on to the secret message
secretMessage += (char)number;
}
System.out.println(secretMessage);
}
}

```

ACTIVITY 11

- 1 $8 \times 640 \times 480 = 2\,457\,600$ bits = 307 200 bytes
- 2 $24 \times 640 \times 480 = 7\,372\,800$ bits = 921 600 bytes

ACTIVITY 12

$44\,100 \times 24 \times 3 \times 60 \times 2 = 381\,024\,000$ bits = 47 628 000 bytes

CHECKPOINT

S1 Text to be converted: 'Data is represented as bits.'

```

01000100 01100001 01110100 01100001 00100000 01101001 01110011 00100000 01110010
01100101 01110000 01110010 01100101 01110011 01100101 01101110 01110100 01100101
01100100 00100000 01100001 01110011 00100000 01100010 01101001 01110100 01110011
00101110

```

S2 $24 \times 640 \times 480 = 737\,280$ bits

C1 How resolution affects image quality is illustrated on page 129.

C2 The factors that affect the file size of audio recordings are listed on page 134 and explained on the previous two pages.

14 DATA STORAGE AND COMPRESSION**ACTIVITY 13**

- 1
 - a 1.5 TB = 1536 GB
 - b 1.5 TB = 1 572 864 MB
 - c 1.5 TB = 1 610 612 736 KB
 - d 1.5 TB = 1.5 TB (given in the question. TB means terabytes)

2 363 143 213 bits = 43.2 mebibytes.

ACTIVITY 14

CODE	RLE VERSION	SIZE OF CODED VERSION
wbbbbbbww	1w5b2w	6
wbbbbbbww	1w5b2w	6
wwwbwww	3w1b4w	6
wwwbwww	3w1b4w	6
wwwbwww	3w1b4w	6
wwwbwww	3w1b4w	6
wwwbwww	3w1b4w	6
wwwbwww	3w1b4w	6
64 bytes		48 bytes

ACTIVITY 15

1	VARIABLES	DATA ITEMS
	text	Input string
	newText	The next character in the input string to be evaluated
	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 A4, B2, C1, D5, E3

CHECKPOINT

S1 bit, nibble, byte, KB, MB, GB, TB

S2 2 TiB = 2 199 023 255 552 bytes.

S3 3a2b9a3c3d

File size of original = 20 bytes

File size of encoded version = 10 bytes


C1 A lossless compression algorithm looks for repeat values in an image in order to create a new more compact representation with a smaller file size. The amount of compression that can be achieved using a lossless compression algorithm depends on how many areas of uniform colour an image has. Images with very short runs of different colours – even if the difference is only marginal – don't compress well.

A lossy algorithm would do away with marginal differences in colour in an image and replace them with the same colour value, relying on the fact that the human eye isn't capable of detecting the difference.

C2 Why a compressed audio file often sounds the same as the uncompressed version is explained on pages 142–143.

15 ENCRYPTION

ACTIVITY 16

1 

2 It is harder to decode messages

ACTIVITY 17

'This is a message' = 'Wklv lv d phvvdjh'

ACTIVITY 18

Students' own answers.

ACTIVITY 19

GRQMWPV VGFRNI

ACTIVITY 20

1 SOMOEYUORWETR

2 Edexcel computer science

CHECKPOINT

S1 HEXE WLSYPH FI IRGVCTXIH

VMBL WZACWH TQ MYGJKXEIV

DHBRASO EGYDT UD NPEALET

S2 It is more secure as there are far more possibilities for each letter to be encrypted.

C1 Gold encrypted with +5 = LTQI. LTQI encrypted with -2 = JROG

Gold encrypted with +3 = JROG

UNIT QUESTIONS

1 **a** 101000000

b An overflow has occurred.

2 **a** 00010010

b Can lead to a loss of precision OR divides by a power of 2.

c The diagram on page 123 (Figure 3.1) illustrates the steps involved in converting from binary to hex. 11101110 is EE in hex.

3 Pixel is defined in the key term box on page 129.

4 **a** There are 36 pixels and one bit is needed to represent each pixel. Encoding pixel information is explained on pages 130–131.

b Assuming that 0 represents black and 1 represents white, the bit pattern is:

100011

101111

100111

101111

101111

100011

c 4 bits would be needed to represent 16 (2^4) colours.

5 a The process of digital recording is explained on page 132.

b Increasing the sampling rate will improve the fidelity of the recording (definition on page 133) and increase the file size of the recording.

c $44\,100 \times 16 \times 2 \times 60 = 84\,672\,000$ bits = 10 584 000 bytes.

6 bit, nibble, byte, MiB, GiB, TiB

7 3w3b2w6b3w3b3w1b

8 'Computer' → 'lusvazkx'

9 'Pzfbkzb' → The unbreakable cipher

10 'The unbreakable cipher'

11 'BEERMASIGHRNO' → 'bring me a horse'

12 a 149

b 11001011

13 a 011110010

b F2 if (a) is correct, otherwise give marks if hexadecimal conversion for any answer to (a) is correct.

14 a 8 is equal to 2^3 and so all of the bits should be moved 3 places to the left. 0s should be inserted into the 3 rightmost positions.

b 00010110

15 11110010

16 $(1024 \times 2300 \times 24) / 8 / 1024 / 1024$

17 a The number of sound samples that are taken each second.

b Increasing the sampling frequency gives a more accurate reproduction of the analogue wave, as more samples are taken with less time between them.

18 Lossy compression removes some of the data to compress a file. With images we do not notice, but with a text file some of the letters would be missing.

19

CIPHERTEXT	KEY	PLAINTEXT
Cnkxk gxk eua	+6 or right 6	Where are you
Yofkd x elopb	-3 or left 3	Bring a horse

20

PLAINTEXT	L	I	K	E	A	R	O	L	L	I	N	G	S	T	O	N	E
Keystream	S	U	P	R	E	M	E	S	U	P	R	E	M	E	S	U	P
Ciphertext	D	C	Z	V	E	D	S	D	F	X	E	K	E	X	G	H	T

UNIT 4: COMPUTERS

16 MACHINES AND COMPUTATIONAL MODELS

ACTIVITY 1

- 1 **a** Inputs could include keyboard, mouse, scanner, microphone, camera, interactive whiteboard.
Outputs could include screen, printer, speakers, headphones, interactive whiteboard, projector.
- b** Inputs could include sensors (temperature, water level, weight of load, door), touch screen controls, switches.
Outputs could include motor, pump, door lock, heater, detergent mixer.
- 2 Computing devices that a central heating installer might use could include tablet, exhaust temperature probe, portable printer, fault-finding equipment, mobile phone, satnav.
- 3 Categories could include manual input devices, automated input devices, visual output, audio output, actuators, switches, sensors, input/output.

ACTIVITY 2

- 1 Challenge C1 on page 164 is a useful starting point for this activity.
- 2 Input: password.
Outputs: message ('Strong' or 'Weak')
Processing: selects which message to display based on the length of the password and the symbols it contains.

ACTIVITY 3

This is a great activity for revisiting topics covered in the previous chapters, namely ASCII representation of text (pages 125 & 126), logical operators (page 42) and linear searches (page 19).

Here is the program implemented in Python.

```
password = input('Enter your password.')
IF len(password) >= 8:
    length = True
ELSE:
    length = False
upperCase = False

index = 0
WHILE (index < len(password)) AND (upperCase == False):

    IF ord(password[index]) >= 65 AND ord(password[index]) <= 90:
        upperCase = True
```

#checks length

#checks for an upper case character

#uses a WHILE loop, so search finishes when the first upper case letter found – more efficient than a FOR loop

```

ELSE:
    index += 1
permittedSymbols = ['!', '#', '&', '*', '+', '?']
symbol = False
index = 0
WHILE (index < len(password)) AND (symbol == False):
    IF password[index] in permittedSymbols:
        symbol = True
    ELSE:
        index += 1
IF length AND upperCase AND symbol:
    print('Strong')
elif (length and upperCase) OR (length and symbol) OR (upperCase and symbol):
    print('Medium')
ELSE:
    print('Weak')

```

#checks for a symbol

#prints message

Here is the program implemented in C#:

```

Console.WriteLine("Please enter a password:");
string password = Console.ReadLine();
bool containsUpper = false, containsSymbol = false;
for(int i = 0; i < password.Length; i++)
{
    //check if current character is uppercase
    if (Char.IsUpper(password, i))
    {
        containsUpper = true;
    }
    //check if current character is not a letter or digit, if it isn't assume it is a symbol
    else if (!Char.IsLetterOrDigit(password, i))
    {
        containsSymbol = true;
    }
}

if (password.Length >= 8 && containsUpper && containsSymbol)
{
    Console.WriteLine("Strong");
}
else
{
    Console.WriteLine("Weak");
}

```

Here is the program implemented in Java:

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter your password:");
        String password = scan.nextLine();
        scan.close();

        // assume password is weak
        Boolean longEnough = false;
        Boolean containsUpperCase = false;
        Boolean containsSymbol = false;

        // check if password is long enough
        if (password.length() >= 8) {
            longEnough = true;
        }

        // define which characters count as symbols
        String symbols = "!#&*+?";

        // loop through each character in the password
        for(int i = 0; i < password.length(); i++) {
            char character = password.charAt(i);

            // check if the password contains a capital letter
            if(Character.isUpperCase(character)) {
                containsUpperCase = true;
            }

            // check if the password contains a symbol
            if(symbols.indexOf(character) >= 0) {
                containsSymbol = true;
            }
        }

        // display password strength
        if(containsUpperCase && containsSymbol && longEnough) {
            System.out.println("Strong");
        } else {
            System.out.println("Weak");
        }
    }
}
```

CHECKPOINT

S1 The input–process–output model is defined on pages 160–163.

S2 Inputs: login details and other text entered via the keyboard, image captured by a camera/webcam.

Outputs: image displayed on screen.

Processing: The computer would need to authenticate the user's login details, locate the appropriate web page, possibly convert the image to an appropriate file type and optimise it for display on the web, strip out any/all the metadata, upload the image.

S3 The sequential, parallel and multi-agent computational models are defined on pages 162 & 163.

C1 Here is the program implemented in Python, with comments used to identify the inputs, output and processing:

#inputs

```
number1 = int(input('Enter first number:'))
number2 = int(input('Enter second number:'))
number3 = int(input('Enter third number:'))
```

#processing

```
total = number1 + number2 + number3
```

#output

```
print(total)
```

Here is the program implemented in C#:

```
string numberString = "";

// read in three numbers
Console.WriteLine("Enter first number:");
numberString = Console.ReadLine();
int number1 = int.Parse(numberString);

Console.WriteLine("Enter second number:");
numberString = Console.ReadLine();
int number2 = int.Parse(numberString);

Console.WriteLine("Enter third number:");
numberString = Console.ReadLine();
int number3 = int.Parse(numberString);

// work out total
int total = number1 + number2 + number3;

// output
Console.WriteLine("Total is" + total);
```

Here is the program implemented in Java:

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
```

```

// input
System.out.print("Enter first number:");
float firstNumber = scan.nextFloat();
scan.nextLine();

System.out.print("Enter second number:");
float secondNumber = scan.nextFloat();
scan.nextLine();

System.out.print("Enter third number:");
float thirdNumber = scan.nextFloat();
scan.nextLine();
scan.close();

// process
float total = firstNumber + secondNumber + thirdNumber;

// output
System.out.printf("Total: %f \n", total);
}
}

```

17 HARDWARE

ACTIVITY 4

Students to produce a suitable design on the research suggested.

ACTIVITY 5

No solution required.

ACTIVITIES 6 & 7

No solution required. The subject vocabulary box on page 167 and the boxed text on page 172 summarise the fetch–decode–execute cycle.

ACTIVITY 8

A Google search for ARM versus Intel will provide students with plenty of information to complete this activity.

ACTIVITY 9

The important thing for students to learn from this activity is that the term ‘spooling’ means putting jobs (in this case print jobs) in a special area called a buffer, in memory or on a disc, so that slower peripheral devices, such as printers, can work at their own pace without slowing down the processor.

Students may already have come across the memory buffer register (MBR) when researching the fetch–decode–execute cycle in Activity 5. This is another example of a buffer being used as a temporary store for data.

ACTIVITY 10

- 1 The purpose of cache memory is explained on pages 168 & 169.
- 2 The difference between RAM and ROM is explained on pages 167 & 168.

- 3 If more RAM is added then there is less need to use virtual memory which is much slower.
- 4 This is an extension of activities 5 and 6 on page 166.
- 5 The most obvious difference is size – components have got considerably smaller and the Pi has fewer of them than the ZX81. USBs weren't around back in the 1980s either. Not visible on the photo, the ZX81 had a proper keyboard and used a tape recorder for storage. It didn't come with a screen: customers were expected to use their televisions for this. In today's money, the ZX81 cost considerably more than a Raspberry Pi and had only a fraction of the processing power, but like the Pi it did inspire a lot of people to try their hand at programming.

ACTIVITY 11

No solution required.

ACTIVITY 12

A microprocessor with four processor cores. The advantages of multicore processors over single-core processors are that the cores can work together on the same program. This is called parallel processing. Or they can work on different programs at the same time. This is called multitasking.

Cache memory uses fast memory within the CPU. This memory is used to store recently used data and data likely to be frequently used so that it can be rapidly retrieved instead of having to fetch it from the slower main memory.

ACTIVITY 13

- 1 The purpose of secondary storage is explained on pages 175–176.
- 2 Magnetic and solid-state storage are described on page 175.
- 3 There isn't a definitive 'best choice' for any of Mohammad's stated activities. Remind students that they need to explain why their chosen secondary storage device is appropriate. For example, a USB memory stick is extremely portable so might be a good choice for storing garden designs to take out to show customers, providing they have a computer. Alternatively, Mohammad could buy a laptop with solid-state storage, which is more robust than a magnetic hard drive and therefore less likely to be damaged if Mohammad drops the laptop whilst out and about.

4 & 5 No solution required.

ACTIVITY 14

No solution required.

ACTIVITY 15

- 1
 - a The multitude of sensors in an engine management system (EMS) collect data such as throttle position, vehicle speed, engine temperature, engine oil pressure, fuel pressure, oxygen. The EMS uses actuators to control engine components such as the ignition, fuel injector, fuel pump, dashboard display, gear adjustment.
 - b The EMS controls the running of an engine by monitoring the engine speed, load and temperature, providing the ignition spark at the right time and controlling the amount of fuel made available to the engine.
- 2 Embedded systems students are likely to encounter in their daily lives include: TV, remote control, fridge, microwave, washing machine, oven, electronic toys, car, sports band, traffic lights, central heating, burglar alarm, under-floor heating, solar panels, lift.

ACTIVITY 16

- 1 This would make a great Raspberry Pi project. Alternatively, students could investigate the traffic light program available on pythoncode.co.uk. Here's a simple text-based program written in Python. It uses the time library module.

```

import time
from time import sleep

start = 'n'
WHILE start != 'y':
    start = input("Press 'y' to start.")

FOR sequence in range(8):
    light = ['red', 'red and amber', 'green', 'amber'][sequence % 4]    #Light sequence repeated twice
    print (light)
    time.sleep(1.5)

```

Here's a simple text-based program written in C#:

```

String[] sequence = new string[4] {"Red", "Red and Amber", "Green", "Amber"};

foreach(string item in sequence)
{
    Console.WriteLine(item);
    System.Threading.Thread.Sleep(1000);    //delay for 1 second
}

```

Here's a simple text-based program written in Java:

```

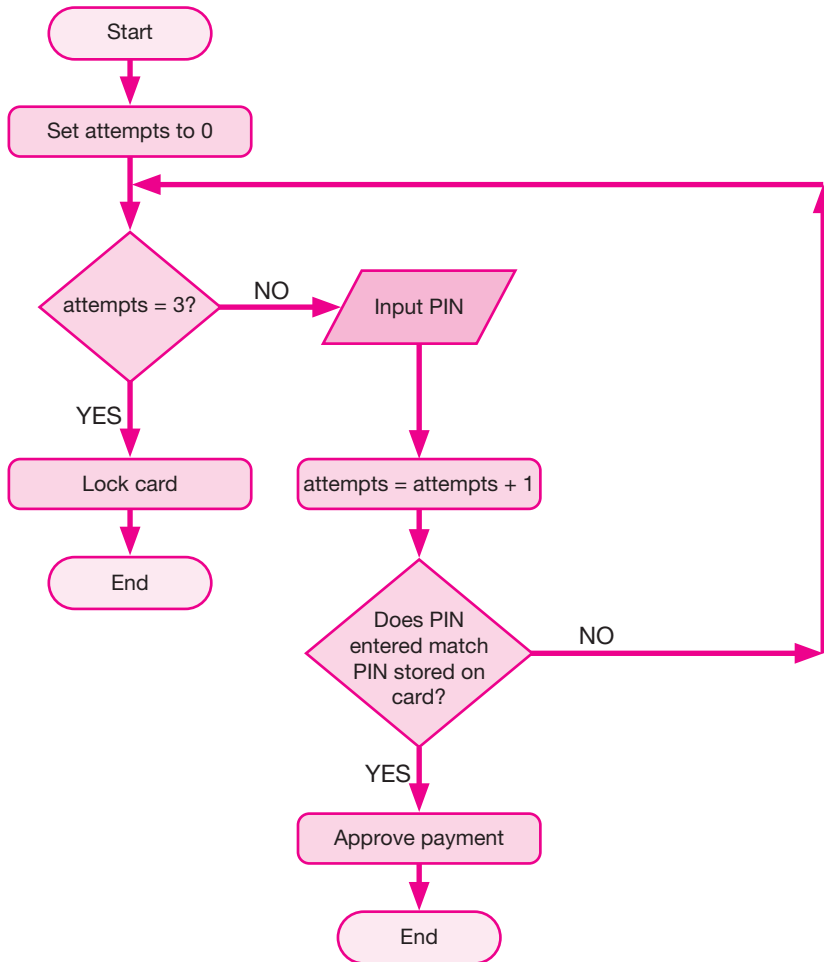
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        // wait until user types y
        String start = "n";
        while(!start.equals("y")) {
            System.out.print("Press 'y' to start:");
            start = scan.nextLine();
        }
        scan.close();

        try {
            // light sequence repeated twice
            String[] lights = {"red", "red and amber", "green", "amber"};
            for(int i = 0; i < 8; i++) {
                System.out.println(lights[i % 4]);
                Thread.sleep(1500);
            }
        } catch (InterruptedException e) {
            System.out.println("Sequence interrupted");
        }
    }
}

```


2



CHECKPOINT

S1 The role of the memory, buses and registers in the fetch–decode–execute cycle is explained on pages 171–172.

S2 Number of cores, size of cache, clock speed.

S3 When comparing different kinds of secondary storage, students should consider factors such as speed, portability, durability and capacity. Points they might make are:

- Data is written to and read from a magnetic hard disc more quickly than to/from an optical disc, so backing up and restoring takes less time.
- Hard discs are permanently located within a drive so are less portable than optical discs, such as DVDs, which can be removed from the drive when not in use. However, nowadays external hard drives are very light and compact so are reasonably portable. That said, an external hard drive still has moving parts, which might get damaged if the drive is roughly handled or dropped.
- In general, optical media tends to be more durable than magnetic media.
- DVDs offer unlimited storage in the sense that you can use as many as are needed. However, nowadays HDDs can store several terabytes of data.
- Magnetic hard drives have now been replaced by solid-state drives in many laptops. Solid-state drives are faster, lighter, quieter and more resilient than magnetic hard drives. They are, however, more expensive and have a smaller capacity than HDDs.

S4 The ‘Internet of things’ is briefly described on page 179 and revisited in Unit 5, pages 210 and 211. Students may want to carry out some additional research if they’re interested in this topic.

C1 The operation of a computer with von Neumann architecture is explained on pages 165 and 166.

- C2** All four are types of storage and their characteristics and function are described on pages 167 and 168 focus on cache, RAM and ROM.
- C3** Cache memory within the processor itself speeds up the processing of data as it stores recently processed data and data frequently used. The data therefore does not have to be fetched from the slower main memory.
- C4** The purpose of secondary storage is explained on pages 175 and 176. Optical storage is ideal for data that doesn't need to be/ mustn't be changed, e.g. films, music, financial records, and archives.
- C5** This challenge requires students to conduct independent research into embedded systems used in driverless cars. They may prefer to focus on existing assisted driving technologies, such as collision avoidance, cruise control or self-parking.

18 LOGIC

ACTIVITY 17

- The game is over either when the player's score exceeds 1 million or if the player runs out of 'health' and isn't operating in 'god mode'.
- ALU
- Here is the 'starter' program written in Python:

```
yearGroup = input('Enter your year group:')
grade = input('Enter your grade (9 - 1):')
target = input('Enter your target grade (9 - 1):')
IF yearGroup == '11' AND grade < target:
    print('\nYou should attend the revision class.')
ELSE:
    print("\nThere's no need for you to attend the revision class.")
```

Here is the 'starter' program written in C#:

```
Console.WriteLine("Enter your year group:");
string yearGroup = Console.ReadLine();

Console.WriteLine("Enter your grade (9 - 1):");
string gradeString = Console.ReadLine();
int grade = int.Parse(gradeString);

Console.WriteLine("Enter your target grade (9 - 1):");
string targetString = Console.ReadLine();
int target = int.Parse(targetString);

if (yearGroup == "11" && grade < target)
{
    Console.WriteLine("You should attend the revision class.");
}
else
{
    Console.WriteLine("There's no need for you to attend the revision class.");
}
```

Here is the ‘starter’ program written in Java:

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        System.out.print("Enter your year group:");
        int yearGroup = scan.nextInt();
        scan.nextLine();

        System.out.print("Enter your grade:");
        int grade = scan.nextInt();
        scan.nextLine();

        System.out.print("Enter your target grade:");
        int target = scan.nextInt();
        scan.nextLine();
        if(yearGroup == 11 && grade < target) {
            System.out.println("You should attend the revision class.");
        } else {
            System.out.println("There's no need for you to attend the revision class.");
        }

        scan.close();
    }
}
```

4 Here is the truth table for the statement `year = 11 AND (grade < target OR grade > 7)`

GRADE < TARGET	TARGET > 7	GRADE < TARGET OR TARGET > 7	YEAR = 11	YEAR = 11 AND (GRADE < TARGET OR TARGET > 7)	REVISION_CLASS
0	0	0	0	0	0
0	0	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	1	0	0	0
1	1	1	1	1	1

ACTIVITY 18

```

1 SET advice TO 'nothing needed'
  IF colour = 'yellow' THEN
    IF leaf tips only THEN
      SET advice TO 'magnesium'
    ELSE
      SET advice TO 'nitrogen'
    END IF
  ELSE
    IF colour = 'brown' AND size = 'small' THEN
      SET advice TO 'phosphorous'
    ELSE
      IF colour = 'brown' AND size = 'normal' THEN
        SET advice TO 'potassium'
      ELSE
        IF leaves = 'cracked' OR leaves = 'misshapen' THEN
          SET advice TO 'magnesium'
        END IF
      END IF
    END IF
  END IF
END IF

```

- 2 Abstraction was explained in Unit 1, pages 24–26. Mohammad’s advice is an example of abstraction because it ignores a lot of the details and focuses on just the essential information needed to solve the problem.

ACTIVITY 19

- 1 The code doesn’t work because the condition of the first **IF** statement is met if it is a Saturday, irrespective of whether it’s term time or holiday. Students can see this for themselves by running this Python version of the pseudocode example given in the worked example.

```

alarm = '7:30'
term_time = input('Term time?')
day = input('Day?')

IF term_time == 'n' OR (day == 'Saturday' OR day == 'Sunday'):
    alarm = '9:00'
ELSE:
    IF term_time == 'y' AND day == 'Saturday':
        alarm = '8:00'

print(alarm)

```

Here is the C# version:

```

string alarm = "7:30", term_time, day;
Console.WriteLine("Term time?");
term_time = Console.ReadLine();

Console.WriteLine("Day?");
day = Console.ReadLine();
if (term_time == "n" || (day == "Saturday" || day == "Sunday"))
{

```

```

        alarm = "9:00";
    }
    else
    {
        if (term_time == "y" && day == "Saturday")
        {
            alarm = "8:00";
        }
    }

    Console.WriteLine(alarm);

```

Here is the Java version:

```

import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        String alarm = "7:30";
        System.out.print("Term time?");
        String termTime = scan.nextLine();
        System.out.print("Day?");
        String day = scan.nextLine();

        if(termTime.equals("n") || (day.equals("Saturday") || day.equals("Sunday"))) {
            alarm = "9:00";
        } else {
            if(termTime.equals("y") && day.equals("Saturday")) {
                alarm = "8:00";
            }
        }
        System.out.println(alarm);

        scan.close();
    }
}

```

2 Here's an amended version of the program that produces the correct outcome in Python.

```

alarm = '7:30'
term_time = input('Term time?')
day = input('Day?')

IF term_time == 'y' AND day == 'Saturday':
    alarm = '8:00'

ELSE:
    if term_time == 'n' OR (day == 'Saturday' OR day == 'Sunday'):
        alarm = '9:00'
print(alarm)

```

Here's an amended version of the program that produces the correct outcome in C#:

```
string alarm = "7:30", term_time, day;

Console.WriteLine("Term time?");
term_time = Console.ReadLine();

Console.WriteLine("Day?");
day = Console.ReadLine();

if (term_time == "y" && day == "Saturday")
{
    alarm = "8:00";
}
else
{
    if (term_time == "n" || (day == "Saturday" || day == "Sunday"))
    {
        alarm = "9:00";
    }
}

Console.WriteLine(alarm);
```

Here's an amended version of the program that produces the correct outcome in Java:

```
import java.util.*;

class Main {
    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);

        String alarm = "7:30";
        System.out.print("Term time?");
        String termTime = scan.nextLine();
        System.out.print("Day?");
        String day = scan.nextLine();

        if (termTime.equals("y") && day.equals("Saturday")) {
            alarm = "8:00";
        } else {
            if (termTime.equals("n") || (day.equals("Saturday") || day.equals("Sunday"))) {
                alarm = "9:00";
            }
        }

        System.out.println(alarm);
        scan.close();
    }
}
```

ACTIVITY 20

- 1 **a** Nothing displayed.
b 'Superstar' displayed.
c Nothing displayed.
- 2 Answer c would be different.

CHECKPOINT

S1 The **AND** table.

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

S2

```
IF sensor_reading = dark OR headlamp_switch = on THEN
    SEND switch_on_signal TO headlamps
END IF
```

S3 No solution required.

C1

A	B	A AND B	C	(A AND B) OR C	P
0	0	0	0	0	0
0	0	0	1	1	1
0	1	0	0	0	0
0	1	0	1	1	1
1	0	0	0	0	0
1	0	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

C2 $P = \text{NOT}(A = B)$

19 SOFTWARE**ACTIVITY 21**

- 1 No solution required.
- 2 'First come first served' and 'Shortest job first' are two well-known scheduling algorithms that students could research.
- 3 No solution required.

ACTIVITY 22

No solution required.

ACTIVITY 23

- 1 A function for simulating throwing a die is provided in Unit 2 on page 82. There are various ways in which this function could be adapted to increase the probability of throwing a six. One way is to generate a wider range of numbers, e.g. `RANDOM(7)`. If the number generated is less than 6, return the number generated, otherwise return 6.
- 2 No solution required.

CHECKPOINT

- S1** Application software performs specific jobs for a user, e.g. word processing, spreadsheet and database creation software. System software is concerned with the running and maintenance of the computer hardware, the running of the application software on the hardware and keeping it running efficiently.
- S2** This activity builds on and reinforces Activities 21 and 22 on page 190.
- S3** Different types of user interfaces for different purposes are described on page 189.
- S4** Anti-virus software is not essential to the operation of the operating system and isn't an application in its own right, but it does do a useful job – hence its categorisation as utility software.
- S5** The advantages and disadvantages of using computer models to predict what might happen in the future are explained on page 192. There's more in Unit 6 about using computer technology to explore the environmental impact of human activities.
- C1** Processing and scheduling are explained on pages 187–188.
- C2** See S2 above.
- C3** This is quite a tough challenge. Students might instead restrict themselves to exploring one or more of the many foxes–rabbits simulation programs available on the web.

20 PROGRAMMING LANGUAGES**ACTIVITY 24**

Tasks for which assembly language is particularly well suited include ones that require hardware specific code, such as device drivers; embedded devices, where the size of the code is important; real-time systems where speed is critical, etc.

ACTIVITY 25

No solution required.

CHECKPOINT

- S1** Machine code is defined in the Subject Vocabulary box on page 194.
- S2** An assembler converts the mnemonics used in assembly language into machine code.
- S3** The need to translate programs written in high-level languages is explained on page 196.
- S4** The Table 4.5 on page 196 compares the two methods of translation. The main challenge that Momina is likely to face has to do with error detection and correction.
- C1** The differences between high-level and low-level language are explained on page 194.
- C2** Table 4.5 on page 196 summarises the factors to consider when choosing between a compiler and an interpreter.

UNIT QUESTIONS

- 1 Outputs could include touch screen, speaker, earphones, vibration motor, light/torch.
Inputs could include touch screen, camera, microphone, motion sensor.
- 2 All are methods of carrying out the instructions in an algorithm. In the sequential model the instructions are followed, step by step, in order, from start to finish.

In the parallel model different parts of the same task are processed by different processors and the results are combined together.

In the multi-agent model separate tasks are processed by different systems. The systems are independent but cooperate through negotiation and coordination.
- 3 There is a maximum rate at which transistors can work and too much heat is generated.
- 4 They might not be able to work in parallel if the algorithm requires sequential processing, i.e. one task requires output from a previous task.
- 5 Volatile memory loses its content when the power is switched off.
- 6 The CPU can access cache memory much more quickly than RAM. Frequently used data and instructions are therefore loaded in chunks from RAM into the cache. This means that the CPU isn't slowed down by having to wait for data transfers from RAM.
- 7 The control unit places the memory address of the next instruction onto the address bus (1) and sends a memory read signal (2). The content of this memory address is placed on the data bus (3). The content of the data bus is copied into a special register in the CPU called the Memory Buffer Register (4).
- 8

```
IF (S = 1 OR B = 1) AND M = 0 THEN
    SEND open TO door_opening_mechanism
END IF
```
- 9

```
SET standardPrice TO 5
IF age < 4 THEN
    SET price TO 0
ELSE
    IF age < 16 OR age > 65 THEN
        SET price TO standardPrice/2
    ELSE
        SET price TO standardPrice
    END IF
END IF

SET standardPrice TO 5
IF age < 4 THEN
    SET price TO 0
ELSE
    IF age < 16 OR age > 65 THEN
        SET price TO standardPrice/2
    ELSE
        SET price TO standardPrice
    END IF
END IF
```
- 10 a The operating system uses scheduling to give each program exclusive use of the CPU for a short time before switching to the next program. This creates the illusion that several applications are running simultaneously.

- b** Functions of an operating system include managing memory – sharing RAM between applications and transferring data and programs in and out of memory; managing files and maintaining a file directory structure; providing a user interface; managing peripheral devices such as disc drives and printers; authenticating users and controlling access to programs and data.
- 11 a** High-level languages use natural language commands such as print and repeat that are easy for humans to understand. However, computers only understand machine code, so Manjit's program will need to be *translated* into that.
- b** Table 4.5 on page 196 summarises the differences.
- c** An assembler does not translate a high-level language into machine code. It is used to translate assembly language.
- 12** Data is entered into the computer and is manipulated according to the instructions of a program. The results of the processing are presented to the user in a suitable form.
- 13 Input:** the user enters their login name and password using a computer.
Processing: the login name entered is checked to ensure that it exists and then the password is checked to ensure it is correct for the login name.
Output: notify the user if they are incorrect and ask them to enter them again.
- 14 a** The function of the CPU is to fetch and execute program instructions stored in memory.
- b** A 2.2 GHz CPU has a clock speed of 2.2 GHz. This gives the number of instructions which can be processed each second as 2.2 billion.
- 15** Program counter, Memory address register, Memory data register, Accumulator.
- 16** The performance cannot be increased indefinitely as the speed that transistors can process data is limited. Also higher speeds generate a greater amount of heat which can cause the CPU to malfunction.
- 17 a** Programs stored in ROM carry out specific tasks including initialising hardware components and starting the operating system when a computer is switched on.
- b** The content of non-volatile memory is not lost when the power is turned off whereas the content of volatile memory is erased.
 The content of volatile memory changes constantly whereas the content of non-volatile memory is fixed and cannot be altered.
- 18** Unlike RAM, which is volatile, secondary storage doesn't lose its content when there is no power. This means that it can provide permanent storage for programs and data.
- 19** The data is not restricted to one location and can be accessed from anywhere in the world with an Internet connection. Many users can access the data and collaborate with each other from anywhere in the world.

20

A	B	P
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	FALSE

21 Interpreter

Advantage: it will stop when it finds an error and so pinpoint it for the developer.

Disadvantage: slower to execute the program as each line has to be translated each time it is run.

Compiler

Advantage: the translation needs to be done only once.

Disadvantage: if the program needs to be changed, then the original source code has to be edited and recompiled.

UNIT 5: COMMUNICATION AND THE INTERNET

21 NETWORKS

ACTIVITY 1

- 1 There's a definition of a network on page 202.
- 2 This will depend on the school or college, but might include printing, Internet access and file storage.
- 3 Services provided by a home network include:
 - access to shared devices such as printers, NAS drives, central heating controllers, etc.
 - access to the Internet
 - file sharing.

ACTIVITY 2

- 1 The difference between a client-server and a peer-to-peer network is explained on page 204.
- 2 One major use of peer-to-peer (P2P) networks is file sharing – notably media files. In a P2P network, each 'peer' is an end-user's computer connected to another 'peer' via the Internet – without going through an intermediary server. To participate, users must download and install P2P software that searches other connected computers on the P2P network to locate specified content.

Most P2P software has file-sharing features that are turned on by default, making any media files on a user's computer available to others to download. As a result, it's perfectly feasible for a user to unwittingly share files stored on their computer.

Policing P2P networks is difficult. No records of file downloads are kept and even if one source of an illegal file is pinpointed and shut down, there are hundreds/thousands of other computers that have the same file installed. Any one of these can share that file with other peers.

ACTIVITY 3

- 1 No solution required.
- 2 No solution required.
- 3 The wireless mesh network topology is a good choice in this situation for a number of reasons:
 - Robustness: network nodes 'talk' directly to each other. A big advantage of this decentralised topology is that it is 'self-healing'. If one node can no longer operate, the others can still communicate with each other, directly or through one or more intermediate nodes.
 - Cost and simplicity: the technology used is relatively cheap and can be simply maintained and extended by users with limited technical expertise.
 - Ease of deployment: they're relatively easy and quick to deploy.
 - Lack of basic communications infrastructure in many developing regions: a wireless mesh network can also support telephony and other services.

ACTIVITY 4

- 1 No solution required.
- 2 Students should look back at the worked example on page 212 to work out how long it would take to download a 300 MB file.

ACTIVITY 5

The purpose of HTTP and the difference between HTTP and HTTPS are explained on page 216. In addition, all the major high-street banks provide customers with information about how their online transactions are kept secure.

ACTIVITY 6

- 1 The term 'protocol' is defined in the subject vocabulary box on page 211.
- 2 All the information about protocols that students need to know in order to produce a table is described on pages 213–216.
- 3 Students should produce a version of the diagram shown in Figure 5.7 on page 217 and there's an explanation of the purpose of each layer on page 214.

CHECKPOINT

- S1** The difference between a LAN and a WAN is explained on page 203.
- S2** The table on page 211 summarises the advantages and disadvantages of wired and wireless networks.
- S3** The three main email protocols, SMTP, POP3 and IMAP, are described on page 213.
- S4** A protocol stack is a collection of protocols that work together. The TCP/IP stack has four layers. It is described on pages 214–215.
- S5** There is a diagram of the 4-layer TCP/IP model on page 217.
- S6** There's a diagram of a bus topology on page 206 (Figure 5.2), a ring topology on page 207 (Figure 5.3), a star topology on page 208 (Figure 5.4) and a mesh topology on page 209 (Figure 5.5).
- S7** 4G also uses IP protocol for voice messages whereas 3G uses it only for data transfer.
- C1** The supermarket's stores are likely to be scattered round the country so connecting them via a WAN would allow them to communicate and share data with head office and with each other.
- C2** There's no central server to 'eavesdrop' on communications in a P2P network. Files are sent directly to the recipient.
- C3** The advantages of using a star network are listed on page 209.
- C4** POP3 creates local copies of emails and deletes the originals from the server, with the result that viewing is restricted to the computer on which they've been downloaded.
- In contrast, IMAP allows users to log into many different email clients or webmail interfaces and view the same emails, because the emails are kept on remote email servers. Furthermore, a user's inbox, sent, and customised folders look alike, and have the same content, whether they're checking mail on their smart phone, tablet, or PC.
- Having instant access to email wherever you happened to be may well be a factor that has contributed to the popularity of smart phones – though there are of course many others.
- C5** See the protocols of the application layer table on page 218.

22 NETWORK SECURITY**ACTIVITY 7**

- 1 This activity gives students a good opportunity to revisit working with text files and subprograms. Here is the solution written in Python.

```
def read_in_data():
    rawData = open("users.txt", "r")
    inputData = rawData.readlines()
    rawData.close()

    users = []
    index = 0
```

```
FOR line in inputData:
    users.append(inputData[index].split(","))
    index += 1
FOR names in users:
    names[1] = names[1].rstrip()
    return users
###end function read_in_data###
def check_user_name():
```

#Splits each line of `rawData` into a list of strings

#Appends each list to the 2-dimensional array `users`
#Strips out new line characters

```
attempt = 1
```

#Returns EITHER the position in the list of a valid username
OR a zero value for 'index' that indicates that the name isn't
in the list.

```
    nameCorrect = False
WHILE attempt < 4 AND nameCorrect == False:
    print('\nUsername attempt:', attempt)
    nameEntered = input('Enter your username:')
    valid = False
    index = 0
    WHILE valid == False AND index < length:
        IF users[index][0] == nameEntered:
            valid = True
        ELSE:
            index += 1
    IF valid == False:
        print('Invalid username.')
        attempt += 1
    ELSE:
        nameCorrect = True
        return index
###end function check_user_name###
```

```
def enter_password(position):
```

```
    attempt = 1
```

#Gives user three goes to enter the correct password

```
    password = ''
    WHILE attempt < 4 and password != users[position][1]:
        print('\nPassword attempt:', attempt)
        password = input('Enter your password:')

    IF password == users[position][1]:
        print('Correct password entered. Proceed.')
    ELSE:
        print('Password incorrect.')
        attempt += 1
    IF attempt == 4:
        print('\nYour account is locked. Contact the system administrator.')
###end function enter_password###
```

```
####main program####
```

```
users = read_in_data()
```

```
length = len(users)
```

```
positionInList = check_user_name()
```

```
IF positionInList < length:
```

```
    enter_password(positionInList)
```

```
ELSE:
```

```
    print('/nContact the system administrator')
```

Here is the solution written in C#:

```

class Program
{
    static void Main(string[] args)
    {
        //declare array
        string[,] userDetails;
        //populate array using Read_in_data function
        userDetails = Read_in_data();

        int attemps = 1;
        bool loggedin = false;
        string username = "";

        while (attemps < 4)
        {
            //get username and password from user
            Console.WriteLine("Enter username:");
            username = Console.ReadLine();
            Console.WriteLine("Enter password:");
            string password = Console.ReadLine();

            //check if username is in array of usernames and passwords
            int usernameIndex = Find_username(userDetails, username);
            if (usernameIndex == 9999)
            {
                //username not found. Increment attemps and go to start of while
                attemps += 1;
                Console.WriteLine("username or password incorrect"); //deliberately vague so user
                                                                    //doesn't know if username
                                                                    //or password wrong

                continue;
            }
            else
            {
                //username exists, so check if password matches
                if (userDetails[usernameIndex,1] == password)
                {
                    //user found
                    loggedin = true;
                    break; // exit loop
                }
                else
                {
                    //password doesn't match
                    attemps += 1;
                    Console.WriteLine("username or password incorrect");
                    continue;
                }
            }
        }
    }
}

```

```

    }
}

    if (loggedin)
    {
        Console.WriteLine("Welcome" + username);
    }
    else
    {
        Console.WriteLine("Too many attempts. Seek assistance.");
    }

    Console.ReadLine();
}

static int Find_username (string[,] arr, string username)
{
    //search array for username and if found return index, otherwise return 9999
    //linear search
    int index = 0, length;
    length = arr.GetLength(0); //get number of rows in array
    while (index < length)
    {
        if (arr[index, 0] == username)
        {
            return index;
        }
        else
        {
            index++;
        }
    }

    //return 9999 as username not found
    return 9999;
}

static string[,] Read_in_data()
{
    //reads in CSV file and populates array. Array is returned to calling function
    string fileName = "authenticate.csv";
    string[] lines = System.IO.File.ReadAllLines(fileName);
    int length = lines.Length;

```

```

//Array is declared as having the correct number of rows from above and 2 columns
string[,] usernameAndPassword = new string[length, 2];

for (int i = 0; i <= (length - 1); i++)
{
    string[] tempLine = lines[i].Split(',');

    usernameAndPassword[i, 0] = tempLine[0];
    usernameAndPassword[i, 1] = tempLine[1];
}

return usernameAndPassword;
}
}

```

Here is the solution written in Java. Java code should usually be object oriented wherever possible but most teachers consider object-oriented solutions to be beyond the scope of International GCSE.

```

import java.util.*;
import java.io.*;

class Main {
    public static String[][] readInData(String fileName) throws IOException {
        FileReader fileReader = new FileReader(fileName);
        BufferedReader bufferedReader = new BufferedReader(fileReader);

        ArrayList<String[]> users = new ArrayList<String[]>();
        String line = bufferedReader.readLine();
        while(line != null) {
            String[] parts = line.split(",");
            users.add(parts);
            line = bufferedReader.readLine();
        }
        bufferedReader.close();
        return users.toArray(new String[users.size()][2]);
    }
}

```

//returns the position in the list of a valid user or -1 if the user isn't found

```

public static int checkUserName(String[][] users, Scanner scan) {
    int attempt = 1;
    Boolean nameCorrect = false;

    while(attempt < 4 && nameCorrect == false) {
        System.out.printf("Username attempt: %d\n", attempt);
        System.out.print("Enter your username:");
        String nameEntered = scan.nextLine();
        Boolean valid = false;
    }
}

```



```
        for(int i = 0; i < users.length; i++) {  
            if(users[i][0].equals(nameEntered)) {  
                valid = true;  
                return i;  
            }  
        }  
    }
```

```
        if(valid) {  
            nameCorrect = true;  
        } else {  
            System.out.println("Invalid user name.");  
            attempt++;  
        }  
    }  
    return -1;  
}
```

// Gives user three goes to enter the correct password

```
public static Boolean enterPassword(int positionInList, String[][] users, Scanner scan) {  
    int attempt = 1;  
    String password = "";
```

// give the user 3 attempts to get their password correct

```
    while(attempt < 4 && !password.equals(users[positionInList][1])) {  
        System.out.printf("Password attempt: %d\n", attempt);  
        System.out.print("Enter your password:");  
        password = scan.nextLine();
```

```
        if(password.equals(users[positionInList][1])){  
            System.out.println("Correct password entered, proceed.");  
            return true;  
        } else {  
            System.out.println("Password incorrect.");  
            attempt++;  
        }  
    }  
    System.out.println("Your account is locked. Contact the system administrator.");  
    return false;  
}
```

```
public static void main(String[] args) {  
    Scanner scan = new Scanner(System.in);  
    try {  
        String[][] users = readInData("users.txt");  
        int positionInList = checkUserName(users, scan);  
        if(positionInList > -1) {  
            enterPassword(positionInList, users, scan);  
        } else {
```

```

        System.out.println("Contact the system administrator.");
    }
    } catch(IOException e) {
        System.err.println("Could not load list of users.");
    }

    System.out.println("done");
    scan.close();
}
}

```

- 2 One simple way of making this program more secure would be to conceal the password that the user enters. Limiting the number of attempts a user has to enter a valid user name and matching password is already a step in the right direction, but letting them have three goes may be not such a good idea.

ACTIVITY 8

- 1 Authentication is explained on page 222. A web search will provide students with plenty of further information about secure passwords. They may want to revisit Activity 7, in which they produced a password checker program.
- 2 The servers should be located in an area where physical access can be controlled, preferably somewhere that is safe from flooding. Obviously, the technicians will need access to the servers – but arguably no one else.

ACTIVITY 9

Factors students should take into account are described on page 225.

ACTIVITY 10

- 1 No solution required.
- 2 No solution required.

ACTIVITY 11

- 1 No solution required.
- 2 Admitting to having been the target of a successful cyber attack could have a negative impact on a business's reputation, with serious financial consequences.

ACTIVITY 12

By not applying a patch, the user increases their chance of becoming a victim of a malware attack that exploits a known flaw in the web browser software in order to do its work.

ACTIVITY 13

- 1 No solution required.
- 2 No solution required.

CHECKPOINT

- S1** Authentication is explained on page 222.
S2 Access control is explained on page 223.
S3 Social engineering is defined on pages 227–228, along with a description of common social engineering techniques.
- C1** Students should refer to page 227 where forms of cyber attack are described.
C2 Reasons might include:

- More cyber attacks are now being reported.
- More small businesses are now operating online and they are often weak target, particularly vulnerable to cybercrime.
- Organised hacking crime groups have replaced individual lone hackers as the main source of cyber attacks. Thus cybercriminals are becoming more expert and using ever more sophisticated methods.
- Because many organisations are unwilling to admit to having been hacked, they don't collaborate sufficiently to fight cybercrime effectively.

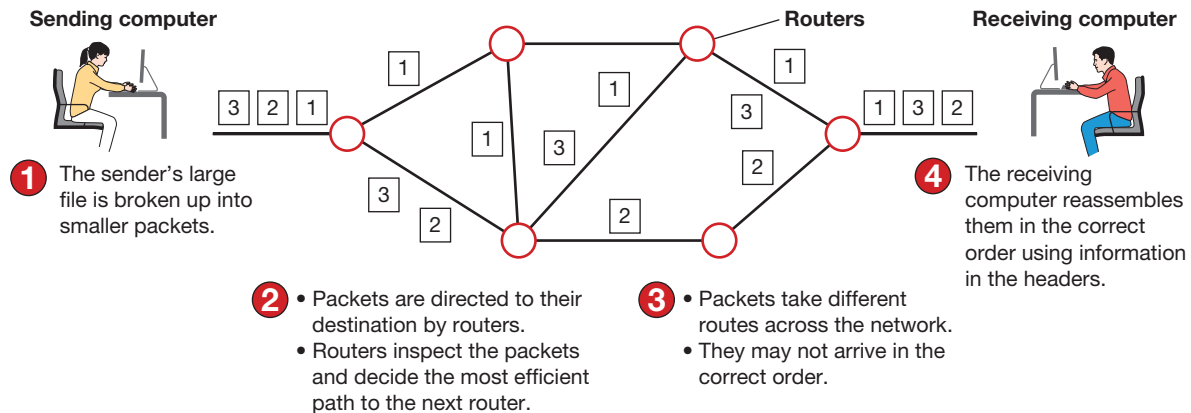
23 THE INTERNET AND THE WORLD WIDE WEB

ACTIVITY 14

Services that use the WWW include messaging, file sharing, video conferencing, e-commerce, media streaming, video gaming, voice calls, software as a service, cloud storage, health monitoring, the Internet of things.

ACTIVITY 15

1 The diagram should illustrate the process described on pages 233 & 234.



2 The Internet is a mesh network. Mesh networks were discussed earlier in this chapter on pages 209 & 210, including what makes them so fault tolerant.

ACTIVITY 16

They will need a switch to connect the computers and other devices to communicate with each other through network cables. Wireless access points will allow users to connect portable devices wirelessly. A router will connect the network to the Internet and a modem will allow the computers to communicate over the Internet.

CHECKPOINT

S1 There's a definition of the Internet on page 233.

S2 The WWW is a service that is provided by the Internet. The Internet itself is a wide area network spanning the globe, incorporating networks in governments, educational establishments and businesses across the world. As the WWW is so widely used, people often refer to it as the Internet, even though they are not actually the same thing.

S3 What happens when a user enters a web address into a browser is explained on page 234, Figure 5.8.

S4 So that all devices from different manufacturers are using the same type of address and can communicate with each other.

C1 Students should include in their explanation HTTP, HTTPS, TCP/IP and HTML. All of these protocols are covered in this chapter.

C2 This is a great topic for class discussion. Factors students may want to consider include:

- the founding principle of the web that anyone, anywhere can use it without paying a fee or having to ask for permission
- its ease of use
- the growth/availability of web-enabled devices
- developments in networking, telecommunications and mobile phone technology
- the dot-com boom of the 1990s
- the growth in popularity of social networking.

C3 A switch connects devices on a local area network by directing communications between them. A wireless access point allows devices to communicate on a wired network using radio signals. A router is used to allow devices to communicate to devices on different networks and a modem converts the computer signals so that they can travel over the medium connecting them, e.g. copper or fibre-optic cables.

UNIT QUESTIONS

- 1 A LAN is confined to just one site, whereas a WAN covers a much larger geographical area. This is essential for the bank since its branches are likely to be scattered across the country.
- 2 The purposes of the layers are as follows:
 - Application layer is where apps such as web browsers and email clients operate.
 - Transport layer sets up the communications between the client and the host, divides the data into packets and checks that packets have been received safely.
 - Data link layer is concerned with transmitting the data through the local area network using protocols such as Ethernet or Wi-Fi.
- 3 A version of the star network diagram (Figure 5.4) on page 208 showing three computers (1), a printer (1), a hub/switch (1) and a server (1) is required.
- 4 Phishing is explained on page 227. Students may want to amplify their answers by conducting a Google search for email + phishing + bank.
- 5 C – IMAP
- 6 The Internet is a global system that enables multiple computers to connect to each other – a network of networks. Many different kinds of data are transferred over this network and many applications make use of this system. The World Wide Web (WWW) is one of them.
- 7
 - a
 - (i) Every computer accessing the Internet needs its own unique address so that the other computers know where to send any requested data, such as web page details, and can identify the computers communicating with them. This is provided by the IP address which uniquely identifies the computer on the Internet.
 - (ii) 32 bits (4×8 bits).
 - (iii) IPv6 uses 128 binary bits to create a single unique address on the network rather than the 32 bits of IPv4. Therefore, far more unique addresses are available.
 - b When a browser requests access to a host using a domain name, the client computer contacts a server of the domain name service. This server contains a database of domain names and their IP addresses which allows it to look up the domain name and return the IP address to the client. If this server has not stored this particular domain name in its database, then it requests it from other DNS servers. The client computer can then contact the required host using this IP address.
- 8
 - a A wireless access point allows wireless devices to connect to cabled networks.
 - b The router connects two networks – the home network and the Internet link provided by the family's Internet service provider. The router sends requests from each of the home computers to Internet servers and distributes the incoming data to the correct computers identified by their IP addresses.
- 9 Diagram to show server and computers each connected by cable to a switch.

A switch is used to link the computers as cables from each one feed into it and so messages can be transmitted from one to the other.

Switches are 'intelligent' as they can read the destination addresses and send them to only the intended computers rather than to all of them.

They can do this because they build up a table of all of the addresses on the network. This cuts down on network traffic.
- 10
 - a Hypertext mark-up language.
 - b HTTP is the protocol used by web browsers in client computers communicate with web servers.

HTTPS allows for communications between a host and client to be secure by ensuring that all communication between them is encrypted.

UNIT 6: THE BIGGER PICTURE

24 COMPUTING AND THE ENVIRONMENTAL IMPACT OF TECHNOLOGY

ACTIVITY 1

Laptops are designed to be more energy efficient than desktops because they rely on battery power. Factors that contribute to their energy efficiency include:

- a low-voltage/ lower-performance processor that generates less heat and therefore requires less cooling
- components such as the graphics processing unit (GPU) and the network interface controller (NIC) are integrated into the chip that contains the processor rather than each being on a separate chip
- a small, energy-efficient power supply unit (PSU)
- a solid-state hard disc drive for storage
- various power-management features designed to conserve the battery
- rather than having an internal DVD drive, an external drive is plugged in and uses power only as and when required.

ACTIVITY 2

- 1 In 2018 global data centre electricity demand was estimated at 198 TWh (terawatt hours) – representing 1% of the global electricity demand. It is predicted to decrease to 191 TWh in 2012 despite a projected 80% increase in data centre traffic. Data centres use energy to power the huge number of servers they use, and the giant fans needed to cool them down.
- 2 Measures that can be used to make data centres more environmentally friendly include:
 - reducing dependency on energy generated from fossil fuels by making more use of renewable energy
 - locating them in regions where it's cold most of the year, so that the amount of energy needed to cool down them down is significantly reduced. Facebook has a huge data centre in the far north of Sweden close to the Arctic Circle
 - adopting energy-efficiency measures, such as using a hot-aisle/cold-aisle configuration to increase cooling system efficiency; using blanking panels to minimise recirculation of hot air and sealing the floor to prevent cooling losses
 - investing in research to develop a new, less energy-hungry alternative to silicon-based data storage. It has been suggested that graphene has enormous potential to do this
 - ration Internet usage – possibly by imposing a tax on uploading data – and/or educate users so that they behave in a more environmentally responsible fashion. Simply switching from colour to black and white photos when uploading to social media sites would have a significant impact.

ACTIVITY 3

Computers 4 Africa (www.computers4africa.org.uk) and Computer Aid International (www.computeraid.org) are two initiatives that make use of pre-owned computing technology.

ACTIVITY 4

Summary of the environmental impact of computing technology and actions that can be taken to reduce it:

ENVIRONMENTAL IMPACT	ACTIONS
Computer technology consumes massive amount of energy, much of it generated from non-renewable fossil fuels, which is contributing to global warming.	Develop more energy-efficient components/practices and switch from non-renewable to renewable sources of energy.
Disposal of cast-off computing technology in landfill sites, which exposes people living nearby to harmful materials and could result in toxic substances leaking into the ground and polluting the air.	Reuse old computer technology rather than consign it to landfill sites. Manage disposal more effectively so that dangerous materials are disposed of safely and reusable materials are recovered.
Use of hazardous materials in the manufacturing process, which put workers at risk.	Legislate to force producers of computing technology to replace hazardous materials with safer alternatives.
Use of valuable, non-renewable materials in the manufacturing process some of which are already in short supply.	Legislate to force producers to use alternative materials and to ensure that valuable materials are recovered from redundant computing technology and reused.

CHECKPOINT

- S1** The table on page 245 lists some of the hazardous substances used in the manufacture of computing technology.
- S2** The danger of dumping e-waste in landfill sites is explained on page 247.
- S3** See solution to Activity 2.
- S4** Ways in which computing technology is helping to preserve the environment are described on page 248.
- C1** The health risks associated with raw material extraction and the manufacture of computing technology components are explained on pages 244 and 245 and those associated with disposal on page 247.
- C2** The use of computing technology to make buildings more energy efficient is outlined on page 248. Other ways that computing technology can help to reduce energy consumption include:
- use of smart meters, thermostats, and sensors – part of the ‘Internet of things’ – to monitor energy
 - raising ‘energy awareness’ and reducing ‘energy wastage’
 - vehicle technologies such as adaptive cruise control that significantly reduce a vehicle’s fuel consumption and emissions
 - use of GPS tracking, wireless communication and adaptive traffic control systems to ease traffic congestion and improve journey efficiency, which – in turn – reduces fuel consumption
 - use of automation and process-control systems to make manufacturing processes more energy efficient
 - modifying the design of hardware to make computer technology more energy efficient
 - writing energy-efficient sorting and searching algorithms.

25 PRIVACY

ACTIVITY 5

No solution required.

ACTIVITY 6

More data means more information, which can be used to create more targeted, focused and efficient services. Examples of how society is benefiting from big data analysis include:

- In retail, companies are analysing large volumes of customer data to provide a smarter shopping experience. This could be through location-based promotions or targeted advertising, or by making the supply chain more efficient.
- In healthcare, big data analytics is being used to predict outbreaks of diseases such as dengue and malaria.
- Disaster-relief organisations are using big data analysis to extract insights and key trends. This provides faster relief and helps staff on the ground to solve problems before they escalate into a crisis.
- The Kepler telescope captures data on 200 000 stars every 30 seconds. Analysis of this mountain of data has led to the discovery of the first earth-like planets outside our solar system.

ACTIVITY 7

Obvious situations in which an invasion of privacy may be justified are those associated with keeping society safe from crime, improving road safety and protection of key installations such as nuclear power stations. As well as citing two of these, students should explain why their severity outweighs the loss of privacy involved.

ACTIVITY 8

Benefits of location-based services include:

- more efficient route planning and navigation systems that use real-time information to avoid hold-ups and congestion
- people on the move can get relevant information about shops, hotels, restaurants, etc. in the vicinity and see if any of their friends are close by or have left tips about the area
- photographs can be geotagged with the precise longitude and latitude of where they were taken
- parents can monitor their offsprings’ whereabouts, helping to keep them out of danger
- conservationists can monitor animal/bird migration patterns
- the response time of emergency services is improved by the availability of accurate and reliable location information, helping to save lives.

Risks of location-based services include:

- They make it easy for predators to locate and stalk their victims.
- They’re a useful source of personal information for criminals who might be intent on identity theft.

- They can be used to establish when a home owner is most likely to be away from home, i.e. the optimum time to carry out a burglary.
- They can be used to infer other sensitive information about an individual, such as their religious affiliation or political standpoint.

ACTIVITY 9

Details can be stolen such as email addresses, bank details and – in some cases – encrypted credit card details.

There can be a huge public backlash. Customers, not surprisingly, blame the company for not having adequate security measures in place.

Affected customers are put at increased risk of identity theft and should inform their bank and credit card company, monitor account activity closely and check their credit rating to make sure nobody applied for credit in their name.

The company suffers both damage to its reputation and considerable financial loss, as customers opt to take their business elsewhere.

CHECKPOINT

- S1** How personal data ends up stored on third-party databases is outlined on page 250.
- S2** The problems associated with personal data falling into the wrong hands are described on pages 250 and 251.
- S3** The table on page 253 provides an overview of privacy-enhancing tools.
- C1** The benefits of revealing personal information are outlined on page 251.
- C2** See solution to Activity 7 and the paragraph on big data on pages 251–252.

26 DIGITAL INCLUSION

ACTIVITY 10

Some useful starting points are:

- 'How mobile phones are changing the developing world', in a series of blog posts published by UNICEF Innovation: <https://blogs.unicef.org/innovation/how-mobile-phones-are-changing-the-developing-world/>
- 'Emerging Nations Embrace Internet, Mobile Technology', written by the Pew Research Center: <http://www.pewglobal.org/2014/02/13/emerging-nations-embrace-Internet-mobile-technology/>

ACTIVITY 11

Actions a government can take to reduce digital exclusion include:

- improving high-speed broadband connectivity so that everyone has access to the Internet irrespective of where they live
- making connection and access to the Internet affordable for all
- providing training courses aimed at helping people to develop digital skills
- making computing a national curriculum subject which all pupils must study
- providing computer/Internet access points in public locations, such as libraries, community centres, cafes and pubs
- making government services, such as car registration and licensing, TV licence renewal and tax returns, available online and encouraging people to use them.

CHECKPOINT

- S1 Technology empowered:** Having affordable access to computing technology and the necessary skills to take advantage of it.

Technology excluded: Not having access to computing technology and/or the skills to use it.

- S2** The table on page 255 highlights some of the drawbacks of being technology-excluded.
- S3** Factors contributing to the digital divide are listed on page 255.
- S4** The solution to Activity 10 illustrates one way of achieving connectivity in regions that have a poor telecoms infrastructure. And the 'Did you know?' box near Activity 10 flags up plans by Facebook to use drones and satellites to overcome the problem.
- C1** See the solution to Activity 11.

27 PROFESSIONALISM

ACTIVITY 12

No solution required.

CHECKPOINT

- S1** Relevant aspects of the BCS Code of Conduct are listed on page 258.
- C1** Professionalism in the context of computer science is discussed on page 258.

28 COMPUTING AND THE LEGAL IMPACT OF TECHNOLOGY

ACTIVITY 13

- 1** No solution required.
- 2** Copyright and patents are explained on page 260. Students may want to include an overview of licensing (explained on page 261) in their podcast.

ACTIVITY 14

PROPRIETARY SOFTWARE	OPEN-SOURCE SOFTWARE
User licences apply strict conditions on the way in which the software is used and distributed.	Under the licence, users can pass on the software to other users for no charge.
The source code is protected. Users aren't allowed to modify it.	Users can study the source code to see how the software works and modify it however they like.
The software is developed professionally and extensively tested prior to release. Any bugs that come to light thereafter are quickly fixed.	The software may not appear as professional or have such a user-friendly interface. It's often released before it has been thoroughly tested so bugs may well have slipped through the net.
Support is provided to keep customers happy so that they will keep using the software. Support and updates may be expensive.	There might be little or no technical support available, but there's likely to be a community of dedicated enthusiasts who are willing to provide help and support free of charge.
Software is developed for the majority of users and may not meet individual needs.	The software can be modified to meet a specific need.
The software must be paid for.	The software is free to use.
The software has been rigorously tested and is usually less likely to have any technical weaknesses.	Criminals may be able to exploit vulnerabilities in the code.

CHECKPOINT

- S1** Software licensing is explained on page 261.
- S2** The difference between open-source and proprietary software is outlined on page 262.
- C1** The protection provided by a patent is explained on page 260.
- C2** Relevant legislation will vary by country, so you will need to conduct some research here. In the UK, laws that computer scientists should be familiar with include the Data Protection Act (1998), the Computer Misuse Act (1990),

the Copyright, Design and Patents Act (1988) and the Regulation of Investigatory Powers Act (2000). Students would also need to look for any amendments to these laws or for relevant new ones.

29 CURRENT AND EMERGING TRENDS

ACTIVITY 15

Students are to design a poster showing items on page 265. Students can research and add other relevant items.

ACTIVITY 16

There are 4 code letters: A, T, G and C. Each codon consists of 3 of them.

Therefore, there are $4^3 = 64$ possibilities.

ACTIVITY 17

Students to produce a suitable design illustrating the uses listed on page 267 and any others using their own research.

ACTIVITY 18

Students to produce a presentation using suitable software, e.g. Powerpoint. Explanation on page 268.

CHECKPOINT

- S1** The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. Intelligent beings are those that can adapt to changing circumstances.
- S2** Because DNA consists of 4 digits which are arranged in groups of 3, it can encode information represented by the bits and bytes of computer systems.
- S3** Nanotechnology is the manipulation of matter with a size from 1 to 100 nanometres.
- C1** Definitions are given on pages 268 & 269.
- C2** Each qubit can be 1 and 0 at the same time and so can calculate a vast number of possible outcomes simultaneously.

UNIT QUESTIONS

- 1** This is an extended answer question. The indicative content that students should include in their response is listed in the summary on page 249. However, emphasise to students that a bulleted list is not an appropriate response to this type of question. In the Paper 1 exam, candidates will be expected to produce a well-structured, essay-style response to this sort of question and use appropriate evidence to support their conclusions.
- 2** The BCS Code of Conduct for computer scientists stipulates that they should not withhold information on the performance of systems. Therefore, the programmer should inform their manager immediately. Furthermore, the code also states that they must avoid injuring others. If the testing software is producing faulty information about exhaust emissions it could also endanger human health, which is another reason for the programmer to take action to flag up the problem.
- 3** A patent gives the patent holder the exclusive right for 20 years to make, use and sell an invention. This encourages inventiveness by ensuring that the owner of the patent (usually the employer of the inventors) gets recognition and benefits financially from the invention. However, in recent years, big companies such as Apple and Samsung have been embroiled in long and expensive legal battles over alleged patent infringements. To defend a patent is very costly. There is an argument that the money spent on legal fees would be better invested in research and development. Patent law encourages companies to keep new inventions secret and block others from using them for 20 years. If inventions were shared from the outset, the pace of technological progress and innovation would be accelerated.
- 4** Any suitable answer, such as: voice recognition, translation, automatic pilots, medical diagnosis. The answer must contain an activity in which the device can improve performance without the need for reprogramming.
- 5 a** The users of devices should consider the implications of continually upgrading or changing them because of the effects on the environment. The manufacture and disposal of the devices consumes large amounts of energy. Most of this energy is derived from non-renewable resources such as fossil fuels, which has an impact on the environment and contributes to global warming.

The devices can be harmful if they are sent to landfill sites for disposal, because toxic waste substances (lead, mercury and cobalt) can get into the land and water. It can be argued that continually changing devices can be ethical because it provides employment for the people making them. Old equipment can be given to people around the world who are not able to afford their own.

The legal responsibilities of users are to recycle their equipment using an official recycling firm. Devices may be dumped in developing countries. It is the ethical responsibility of users to check what happens to them and ensure that they are recycled by reputable companies.

- b** File sharing is unethical as it deprives the creators of the work of payment for their work. If a user obtains a copy of someone's work without paying for it, it is the same as stealing or shoplifting. Without payment, the professional artists will not be able to continue creating their works. It has been estimated that about 50% of jobs in the film industry have been lost by the copying and sharing of videos. Peer-to-peer sharing therefore creates unemployment. File sharing is illegal as it contravenes the Copyright Designs and Patents Act 1988. Sanctions for breaking the Copyright Designs and Patents Act 1988 include being fined or being prevented from using the Internet.
- c** Using another person's password to gain unauthorised access to a computer or a network is illegal under the Computer Misuse Acts of many countries, e.g. European countries, Australia and Singapore and GCC countries, such as Bahrain, Oman, Saudi Arabia or the UAE.

It is also unethical. The student is trying to gain an unfair advantage over fellow students by looking at the past papers and model answers. They are not behaving in a way in which individuals and society thinks of as reflecting good values.

6 Any two from:

- They may not be able to afford equipment.
- They may have low IT literacy.
- There may not be the infrastructure available to them e.g. no broadband.

7 The answer could include some of the following ideas:

- **Energy:** Manufacture and use of devices uses energy. Manufacturing involves energy-intensive mining and processing of minerals. The use of devices involves the energy used by the devices themselves, but also by data centres. These data centres generate heat, so energy is needed to keep them cool.

Much of the energy used comes from non-renewable sources such as gas and coal.

Computer science is used in efficient energy production. Computer software is used to design, model and test efficient devices to produce electricity from wind, wave and solar power. Energy use can be reduced using smart technologies, such as light-sensitive switches that turn off lights when they are not needed.

Efficient transport planning using computer modelling and analysis can reduce fuel use.

- **Sustainability:** Digital devices use many different chemical elements. Some of these are rare and will be in short supply as they are used up. It is difficult to recycle devices to reuse these elements.
- **Waste:** Electronic devices are difficult to recycle and are often disposed of in landfill sites as e-waste. Landfill sites take up areas of land that could be used for other purposes. Toxic substances such as lead, mercury and cobalt can get into the soil and the water supply from the landfill sites and so cause health problems.
- **Data analysis:** Computer science technology can be used to monitor environmental factors by transmitting and analysing data. This data can be shared by scientists around the world who can collaborate to find solutions to problems. Computers can be used to develop models to forecast environmental behaviour and identify options for action.

8 To give the public permission to share and use their work. To allow others to modify and change the original work.

9 a Proprietary software is commercially produced by an organisation for a profit.

b Any one from:

- Software is developed professionally and carefully tested.
- Support will be provided to keep customers happy so that they will continue to use the software.
- There will be books, magazine articles and online tutorials.
- Updates and bug fixes meet the needs and suggestions of the users.

10 Any two from:

- As long as there are cellular organisms, there will always be a supply of DNA.
- The large supply of DNA makes it a cheap resource.
- Unlike the toxic materials used to make traditional microprocessors, DNA biochips can be made cleanly.
- DNA computers are many times smaller than today's computers.

11 A bit is a binary unit used by classical computers to encode data and information.

A bit can be either 1 or 0.

A qubit is used in quantum computers to encode data and information.

A qubit can be both 1 and 0 at the same time until it is observed.