

Edexcel

OL IGCSE

Programming

CODE: (4CP0)

Unit 02



Chapter 05 – Develop code

Once an algorithm has been developed to solve a particular problem, it has to be coded into the programming language that the developer is using. This usually means that the written descriptions, flowcharts and pseudocode have to be converted into actual programming code.

Algorithm and programs

As you learnt in Unit 1, an algorithm is a precise method of solving a problem. It consists of a sequence of unambiguous, step-by-step instructions. A program is an algorithm that has been converted into program code so that it can be **executed** by a computer. A well-written algorithm should be free of logic errors and easy to code in any high-level language.

Data types

As you learnt in Unit 1, algorithms use variables (named memory locations) to store values. 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.

DATA TYPE	DESCRIPTION	EXAMPLE	EXAMPLES OF USE
integer	Used to store whole numbers without a fractional part	30	age = 30 number = 5
real or float	Used to store numbers with a fractional part (decimal place). Real numbers are sometimes referred to as floats (short for floating point)	25.5	weight = 25.5 price = 12.55
Boolean	Only has two possible values: True or False	False	correct = False lightOn = True
character*	A character can be a single letter, a symbol, a number or even a space. It is one of the four basic data types	'm'	gender = 'm' char = ':'
string	A set of characters which can include spaces and numbers and are treated as text rather than numbers	'the computer'	name = 'Catherine' type = 'liquid'

*Python does not have a character data type.

▲ Table 2.1 Common data types

When writing pseudocode, you don't have to specify the data types of variables. However, data types become much more important once you start programming in a high-level language. This is because the data type of a variable determines the operations that can be performed on it.

VARIABLE INITIALISATION

When a variable is declared, the computer gives it a location in its memory. Initially, this location is empty, so before a variable can be used it has to be given a value.

SUBJECT VOCABULARY

execution the process by which a computer carries out the instructions of a computer program

SUBJECT VOCABULARY

initialise to set variables to their starting values at the beginning of a program or subprogram

initialisation the process of assigning an initial value to a variable

assignment statement the SET...TO command is used to initialise variables in pseudocode, for example:

```
SET anotherGo TO 0
```

```
SET correct TO False
```

You can put an initial value into a variable by:

- initialising it when the program is run (e.g. `SET total TO 0`, in pseudocode)
- reading a value from a keyboard or other device (e.g. `RECEIVE admissionCharge FROM (INTEGER) KEYBOARD`).

Once a variable has been **initialised** an **assignment statement** is used to change its value (e.g. `SET total TO total + admissionCharge`).

In Python this would be:

```
total = 0
total = total + admissionCharge
```

In Java this would be:

```
Scanner scan = new Scanner(System.in);
int admissionCharge = scan.nextInt();
int total = 0;
total = total + admissionCharge;
```

In C#, variables must be declared before use. When you declare a variable, you need to state the data type that the variable will store, for example:

```
// declare a variable called total that will be used to
store floating point numbers and assign the value 0.0 to it
float total = 0.0;
// add the value stored in the variable admissionsCharge
to the total
total = total + admissionsCharge;
```

If a variable, such as a loop counter, is intended to hold a running total, then it should always be initialised to a starting value. Some programming languages won't execute if the programmer fails to do this; others will do so but may well produce some unexpected results.

TYPE COERCION

Sometimes the data type of a variable gets changed during program execution. This is known as type coercion. For example, if an integer value and a real value are used in an arithmetic operation, the result will always be a real.

SUBJECT VOCABULARY

type coercion the process of converting the value stored in a variable from one data type to another

Command sequence, selection and iteration

In Unit 1 you learnt that the three key building blocks of algorithms are command sequence, selection and iteration. In this unit, you will have the opportunity to implement these constructs in the high-level programming language you are studying.

SELECTION

The selection construct is used to create a branch in a program. The computer selects which branch to follow based on the outcome of a condition, using an **IF...THEN...ELSE** statement. For example, in pseudocode:

```
IF day = 'Saturday' OR day = 'Sunday' THEN
    SET alarm TO 11
ELSE
    SET alarm TO 8
END IF
```

A standard **IF...THEN...ELSE** statement provides two alternatives. If there are more than two, then in Pearson Edexcel pseudocode a nested **IF** must be used. However, many high-level programming languages have an additional built-in selection construct that does away with the need for a **nested IF statement**.

RELATIONAL OPERATORS

The relational operators are used to compare two values and in Python, Java and C# are all the same.

SUBJECT VOCABULARY

nested IF statement a nested IF statement consists of one or more IF statements placed inside each other. A nested IF is used where there are more than two possible courses of action

SUBJECT VOCABULARY

relational operator an operator that tests the relationship between two entities
logical operator a Boolean operator using AND, OR and NOT

RELATIONAL OPERATOR	PYTHON, JAVA, C#
Equal to	==
Greater than	>
Greater than or equal to	>=
Less than	<
Less than or equal to	<=
Not equal to	!=

LOGICAL OPERATORS

AND

If two conditions are joined by the **AND** operator, then they must both be true for the whole statement to be true.

OR

If two conditions are joined by the **OR** operator, then either one must be true for the whole statement to be true.

NOT

The **NOT** operator reverses the logic of the **AND** and **OR** statements. The statement **IF A = 3 AND B = 6** will be true only if the conditions are met, i.e. A and B are both equal to the values stated.

The statement **IF NOT (A = 3 AND B = 6)** will be true whenever both A and B are NOT equal to the values stated, i.e. either or both are not equal to those values.

Logical operators in high-level languages.

LOGICAL OPERATOR	PYTHON	JAVA	C#
AND	and	&	&&
OR	or		
NOT	not	!	!

LOOPS

A loop is another name for an iteration. Loops are used to make a computer repeat a set of instructions more than once. There are two types of loop: definite and indefinite.

DEFINITE ITERATION

This is used when the number of iterations, or turns of the loop, is known in advance.

In the Pearson Edexcel pseudocode there are two ways of doing this using **REPEAT...END REPEAT** and **FOR...END FOR**.

An example of a **REPEAT** loop is shown below.

```
REPEAT 50 TIMES
  SEND '*' TO DISPLAY
END REPEAT
```

Using a **FOR** loop, this would be:

```
FOR times FROM 1 TO 100 DO
  SEND '*' TO DISPLAY
END FOR
```

FOR loops can also include a step so that the counting is not consecutive. A step is included in the following pseudocode example.

```
FOR times FROM 0 TO 100 STEP 25 DO
  SEND times TO DISPLAY
END FOR
```

The output would be 0, 25, 50, 75.

A definite loop is used when you know in advance how often the instructions in the body of the loop are to be repeated.

NESTED LOOPS

A nested loop is made of a loop within a loop. When one loop is nested within another, each iteration of the outer loop causes the inner loop to be executed until completion.

INDEFINITE ITERATION

An indefinite loop is used when the number of times a loop will need to be repeated is not known in advance. For example, if you want to give a user the option of playing a game as often as they want. Indefinite loops are repeated until a specified condition is reached.

Python

For indefinite iteration, Python uses the 'while' loop – something is done while a condition is met. The following program asks a user to enter a number while the number is less than 20

```
number = 1
while number <= 20:
    number = int(input('Please enter a number'))
print('You entered a number greater than 20')
```

As soon as the number is greater than 20, the program breaks out of the loop and prints a message for the user.

Java

For indefinite iteration, Java uses a 'while' loop – instructions are repeated while a condition is met. The following program asks a user to enter a number while the number is less than 20.

RANDOM NUMBERS

Random numbers are commonly used in games of chance such as flipping a coin or rolling a dice. The aim is to make an event random. All high-level programming languages have functions to create random numbers.

Python

Python has a random module. The following code will generate a random number between 1 and 10.

```
import random
x = random.randint(1, 11)
print(x)
```

The 'import' command is necessary so that the 'random' module can be used.

Java

Java can generate a random integer using `System.util.Random`. The following code will generate a random number between 1 and 10.

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Random r = new Random();
        int x = r.nextInt(10) + 1;
        System.out.println(x);
    }
}
```

Alternatively, you can generate a random real value using `Math.random()`.

```
class Main {
    public static void main(String[] args) {
        int x = (int)(Math.random() * 10) + 1;
        System.out.println(x);
    }
}
```

C#

C# can generate random numbers using the Random Class. The following code will generate a random number between 1 and 10 and display on screen.

```
Random rand = new Random();
randomNumber = rand.Next(1, 11);
Console.WriteLine(randomNumber);
```

Chapter 06 – making programs easy to read

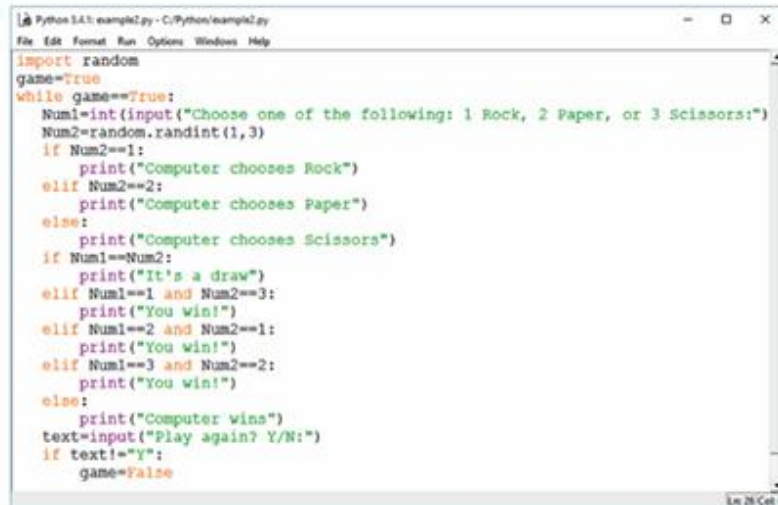
Developers usually work in teams and it is important that they understand how each other's code and programs work, especially if there are errors that need correcting. Code should be written in standard ways and be explained using comments.

Code readability

You should always try to ensure that any code you write is easy to read and understand. We refer to this as 'readability'. This benefits you and anyone else who needs to understand how your programs work. It is surprising how quickly you forget. Try revisiting the programs you have already written in this unit and make sure it is still clear to you what they do and how they work.

Imagine how much more difficult it would be to make sense of a complex program with lots of variables, subprograms, nested loops and multiple selection statements. Programming is not a solo activity. Programmers usually work in teams, with each programmer developing a different part of the program. This only works if they all adopt a standard approach to writing readable code.

► Figure 2.2 An example of Python code



```
import random
game=True
while game==True:
    Num1=int(input("Choose one of the following: 1 Rock, 2 Paper, or 3 Scissors:"))
    Num2=random.randint(1,3)
    if Num2==1:
        print("Computer chooses Rock")
    elif Num2==2:
        print("Computer chooses Paper")
    else:
        print("Computer chooses Scissors")
    if Num1==Num2:
        print("It's a draw")
    elif Num1==1 and Num2==3:
        print("You win!")
    elif Num1==2 and Num2==1:
        print("You win!")
    elif Num1==3 and Num2==2:
        print("You win!")
    else:
        print("Computer wins")
    text=input("Play again? Y/N:")
    if text!="Y":
        game=False
```

The programmer who produced the program in Figure 2.2 did not follow good practice. There are a number of ways that the readability of this code could be improved.

- Use descriptive names for variables (e.g. userChoice instead of Num1).
- Add blank lines between different blocks of code to make them stand out.
- Add comments that explain what each part of the code does.

SUBJECT VOCABULARY

block of code a grouping of two or more code statements

Table 2.5 lists the techniques you should use to make your programs easy to read and understand.

TECHNIQUE	DESCRIPTION
Comments	Comments should be used to explain what each part of the program does.
Descriptive names	Using descriptive identifiers for variables, constants and subprograms helps to make their purpose clear.
Indentation	Indentation makes it easier to see where each block of code starts and finishes. Getting the indentation wrong in Python will result in the program not running or not producing the expected outcomes.
White space	Adding blank lines between different blocks of code makes them stand out.

▲ Table 2.5 Techniques for program clarity

Chapter 7 – Strings

A character is one of the four basic data types. It can be a single letter, a symbol, a number or even a space. A sequence of characters is called a **string**. Although strings can contain different sorts of characters, including numbers, they are all treated as if they were text.

SUBJECT VOCABULARY

string a sequence of characters. They can be letters, numbers, symbols, punctuation marks or spaces

Starting indexing

Each character in a string has an index number, with the first character at position 0. You can use the index to reference individual characters in a string.

Therefore, the index position of the letter 'm' is 2, even though it is the third character, as shown in Table 2.6

INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
String	C	o	m	p	u	t	e	r		S	c	i	e	n	c	e

▲ Table 2.6 An example string index

Length

The Pearson Edexcel pseudocode has a built-in **LENGTH function**, which you can use to find the number of characters in a string. Therefore:

```
SET numChars TO LENGTH(myText)
SEND numChars TO DISPLAY
```

would print '16'. Table 2.7 gives the method for Python, Java and C#.

LANGUAGE	METHOD	EXAMPLE	OUTPUT
Python	The function to find the length of a string is <code>len()</code> .	<code>string = 'Computer Science'</code> <code>length = len(string)</code> <code>print(length)</code>	16
Java	Each string object has a <code>.length()</code> method.	<code>String s = "Computer Science";</code> <code>int length = s.length();</code> <code>System.out.println(length);</code>	16
C#	The length of a string can be found by accessing the length property.	<code>string subject = "Computer Science";</code> <code>Console.WriteLine(subject.Length);</code>	16

▲ Table 2.7 Finding the length of a string in Python, Java and C#.

SUBJECT VOCABULARY

function a subprogram that performs a specific task and returns a value to the main program. High-level programming languages have a number of useful built-in functions. You can also create your own or use functions available in online libraries

String traversal

You can use a **FOR** loop to cycle through each of the characters in a string. This is known as **string traversal**.

The following algorithm prints out the word 'monkey' letter by letter, displaying each letter on a separate line.

```
SET animalName TO 'monkey'
FOR index = 0 TO LENGTH(animalName) - 1
    SEND animalName[index] TO DISPLAY
END FOR
```

The loop runs until `LENGTH(animalName) - 1` because string indexing starts at 0 and therefore has to print characters with indexes of 0 to 5.

PYTHON

```
animalName = 'monkey'
for index in range(0, len(animalName)):
    print(animalName[index])
```

SUBJECT VOCABULARY

string traversal using a loop to cycle through each character in a string

Using Python you should not use `len(animalName) - 1` as the range command does not include the end number of the loop.

JAVA

```
class Main {
    public static void main(String[] args) {
        String animalName = "monkey";
        for(int index = 0; index < animalName.length(); index++) {
            System.out.println(animalName.charAt(index));
        }
    }
}
```

C#

```
string animalName = "monkey";

for (int index = 0; index < animalName.Length; index++)
{
    Console.WriteLine(animalName[index]);
}
```

Other ways to manipulate strings

FINDING A CHARACTER WITH A PARTICULAR INDEX

The index of a character is the position of that character within a string. Note that indexing starts at 0 not 1. This means the first character will have an index of 0 and the second character will have an index of 1.

PYTHON

Python uses square brackets to access elements in a string:

```
string = 'Computer Science'
print(string[11])
```

This code would return the character 'c'.

JAVA

Java allows you to get a character at a particular index using the `charAt` method.

```
class Main {
    public static void main(String[] args) {
        String string = "Computer Science";
        System.out.println(string.charAt(11));
    }
}
```

This code would display the character 'i' (the 12th character at index 11).

C#

```
string text;
text = "Computer Science";
Console.WriteLine(text[3]);
```

This code would return the character 'p'.

CHANGING ALL CHARACTERS TO LOWER CASE

PYTHON

```
string = 'Computer Science'  
string = string.lower()  
print(string)
```

This code would return 'computer science'.

JAVA

```
class Main {  
    public static void main(String[] args) {  
        String string = "Computer Science";  
        string = string.toLowerCase();  
        System.out.println(string);  
    }  
}
```

C#

```
string text;  
text = "Computer Science";  
text = text.ToLower();  
Console.WriteLine(text);
```

Would output 'computer science'.

CHANGING ALL CHARACTERS TO UPPER CASE

PYTHON

```
string = 'Computer Science'  
string = string.upper()  
print(string)
```

This would return 'COMPUTER SCIENCE'.

JAVA

```
class Main {  
    public static void main(String[] args) {  
        String string = "Computer Science";  
        string = string.toUpperCase();  
        System.out.println(string);  
    }  
}
```

C#

```
string text;  
text = "Computer Science";  
text = text.ToUpper();  
Console.WriteLine(text);
```

This would output 'COMPUTER SCIENCE'.

EXTRACTING CHARACTERS FROM A STRING

PYTHON

```
string = 'Computer Science'
substring = string[3:6]
print(substring)
```

This code would return 'put' as it selects the characters from index 3 to index 5. Index 6 is not included.

JAVA

```
class Main {
    public static void main(String[] args) {
        String string = "Computer Science";
        String substring = string.substring(3,6);
        System.out.println(substring);
    }
}
```

C#

```
string text;
text = "Computer Science";
text = text.Substring(3,4);
Console.WriteLine(text);
```

This would display 'pute' as it selects 4 characters starting at index 3 (remember that the first character is at index 0).

CHECKING A PHRASE IN THE STRING

These examples check to see if the string 'Computer Science' contains the strings 'put' and 'PUT': 'put' is a substring of 'Computer Science' but 'PUT' is not, so these examples will display 'True' then 'False'.

PYTHON

```
string = 'Computer Science'
present = 'put' in string
print(present)
present = 'PUT' in string
print(present)
```

JAVA

```
class Main {
    public static void main(String[] args) {
        String string = "Computer Science";
        boolean present = string.contains("put");
        System.out.println(present);
        present = string.contains("PUT");
        System.out.println(present);
    }
}
```

C#

You can check if a substring is present in a string using the contains method. If the substring is present in the string then 'True' is returned, otherwise 'False' is returned.

```
string text;
bool found = false;
text = "Computer Science";
found = text.Contains("put");
Console.WriteLine(found);
```

This would display 'True', as 'put' is in 'Computer Science'.

CONCATENATION

Concatenation involves joining two or more items of information together. Concatenating two strings produces a new string object. It is very useful when displaying text on screen.

Chapter 8 – Data structure

Data should be classified and organised into similar types so it can be easily searched and analysed. They are stored together in data structures, such as records and arrays.

A data structure is an organised collection of related elements. There are many different data structures that can store multiple data items used in programming. You have already encountered one of them – strings. In this section we will investigate two more: arrays and records.

Arrays

An **array** is an organised collection of related values with a single shared identifier. All the elements in an array are the same data type. Each has a unique **index** value denoting its position in the array.

PYTHON	JAVA	C#
<code>len(myArray)</code>	<code>myArray.length</code>	<code>myArray.Length</code>

PYTHON

In Python, arrays are not commonly used. Instead lists are used but they are very similar in the way that they operate. Just like an array, a list is created by adding items, separated by commas, inside square brackets.

JAVA

Arrays are static data structures, which means that you can't change the size of an array to make it bigger or smaller in order to add or remove values.

SUBJECT VOCABULARY

data structure an organised collection of related elements. Arrays and records are two common data structures used in programming

SUBJECT VOCABULARY

array a structure that contains many items of data of the same type. The data is indexed so that a particular item of data can be easily found

index a number that identifies each element of an array in Python and Java

Lists are easier to use as they are **dynamic** – they do not have a fixed size and can grow as new elements are added. When they are declared they do not have to be given a size, e.g.:

```
cars = []
```

When adding items to an array the **append** command is used, for example:

```
cars = []
cars.append('Audi')
```

```
print(cars)
```

would return the following:

```
['Audi']
```

Another advantage of lists is that the items do not have to be of the same data type.

```
cars = []
cars.append('Audi')
cars.append(3)
print(cars)
```

would return:

```
['Audi', 3]
```

There are many functions in Python for manipulating arrays.

Investigate the following:

```
max()
min()
slice()
```

C#

C# supports lists and arrays. The code below uses an array to hold the items.

```
string[] make = { "Ford", "Mercedes", "Toyota", "BMW",
"Audi", "Renault" };

foreach (string currentItem in make)
{
    Console.WriteLine(currentItem);
}
```

MULTIDIMENSIONAL ARRAYS

In the array `cars = ['Ford', 'Mercedes', 'Toyota', 'BMW', 'Audi', 'Renault']` there is only one item at each index position: the name of the manufacturer.

A multidimensional array is an 'array of arrays'; each item at an index is another array.

We declared the above array using this statement, `array cars[6]`.

If we wanted to create a two-dimensional array we could declare it as `array cars[3, 2]` so that there is another array at each index to store two items of information.

Here is an extract from an array named `examResults`. It has three rows, each of which stores a set of four exam results. The mark of 47 is located at `examResults [1, 2]` – second row, third element along – and the value 80 at `examResults [0, 0]` – first row, first column.

	0	1	2	3
0	80	59	34	89
1	31	11	47	64
2	29	56	13	91

▲ Table 2.8 `examResults` two-dimensional array

Each item of data has two indexes. An array to hold this data would be declared as `array[3, 4]`.

If the array was printed it would be `[[80, 59, 34, 89], [31, 11, 47, 64], [29, 56, 13, 91]]`. There are square brackets around each set of results and around the whole array.

In Python, the array would be declared and initialised as:

```
Scores = [[80, 59, 34, 89], [31, 11, 47, 64], [29, 56, 13, 91]]
```

In Java, it would be:

```
int[][] Scores = new int[][] {
    {80, 59, 34, 89},
    {31, 11, 47, 64},
    {29, 56, 13, 91}};
```

In C# the array would be declared and initialised as:

```
int[,] Scores= new int[,] { { 80, 59, 34, 89 }, { 31, 11,
47, 64 }, { 29, 56, 13, 91 } };
```

Records

We have already said that the elements of an array must all be the same data type. In contrast, the record data structure stores a set of related values of different data types.

Each element in a record is known as a **field** and is referenced using a field name.

learnerNum	firstName	lastName	AGE	FORM
1	Isla	Smith	15	10H
2	Shinji	Fujita	14	10B
3	Anita	Khan	15	10A
4	Abdur	Rahman	15	10G

▲ Table 2.10 Example of record data structure

SUBJECT VOCABULARY

record a data structure that stores a set of related values of different data types
field an individual element in a record

Chapter 09 Input / Output

User input

Most programs require some form of input either from a user or from a file. You already know how to receive user input from a keyboard. A program can be made much more 'user friendly' by displaying helpful messages informing users of what they are expected to enter and confirming successful input.

Validation

It is important to ensure that data entered by the user is valid, as invalid data can cause a program to behave unexpectedly or even stop altogether. If the data entered into a program is incorrect, the output it produces will also be wrong. This is sometimes called the Garbage In, Garbage Out (GIGO) principle.

Any program that requires data entry should have appropriate forms of validation built in. But **validation** can't guarantee that the data entered is correct. It can only make sure that it is reasonable.

RANGE CHECK

A range check is used to ensure that the data entered is within a specified range. Study this algorithm, written in Pearson Edexcel pseudocode which checks that the number entered is between 1 and 10.

```

BOOLEAN valid
SET validNum TO False
WHILE validNum = False DO
    SEND 'Please enter a number between 1 and 10:' TO
    DISPLAY
    RECEIVE number FROM (INTEGER) KEYBOARD
    IF number >= 1 AND number <= 10 THEN
        SET validNum TO True
    END IF
END WHILE
SEND 'You have entered:' & number TO DISPLAY

```

PRESENCE CHECK

Another type of validation is a presence check. This simply ensures that a value has been entered, preventing the user from leaving an input blank.

This algorithm asks the user to input their name and uses a presence check to ensure they have entered a value. Any value will cause the loop to finish. It will keep asking the user to input their name until they input a value.

```
SET userName TO ' '
WHILE userName = ' ' DO
    RECEIVE userName FROM (STRING) KEYBOARD
END WHILE
SEND 'Hello' & userName TO DISPLAY
```

LOOK-UP CHECK

A look-up check is used to test that a value is one of a **predefined** set of acceptable values. The list of acceptable values can be stored in a one-dimensional array.

This algorithm stores a list of valid form names in an array. It compares the form name entered by the user with the values in the array.

```
SET arrayForms TO ['7AXB', '7PDB', '7ARL', '7JEH']
RECEIVE form FROM (STRING) KEYBOARD
SET valid TO False
SET index TO 0
SET length TO LENGTH(arrayForms)
WHILE valid = False AND index < length DO
    IF form = arrayForms[index] THEN
        SET valid TO True
    END IF
    SET index TO index + 1
END WHILE

IF valid = True THEN
    SEND 'Valid form' TO DISPLAY
ELSE
    SEND 'The form you entered does not exist.' TO DISPLAY
END IF
```

LENGTH CHECK

It is sometimes necessary to check that the length of a value entered falls within a specified range. For example, all UK postcodes are between six and eight characters long, so validation could be used to check that the length of a postcode entered is within this range. Needless to say, that doesn't necessarily mean it is correct.

```
RECEIVE postCode FROM (STRING) KEYBOARD
SET length TO LENGTH(postCode)
IF length >= 6 AND length <= 8 THEN
    SEND 'Valid' TO DISPLAY
ELSE
    SEND 'Invalid' TO DISPLAY
END IF
```

Testing validation rules

Normal data – This is data that is within the limits of what should be accepted by the program. For example, a password with seven characters fulfils the validation rule that states that passwords must be between six and eight characters in length.

Boundary data – This is data that is at the outer limits of what should be accepted by the program. For example, if a validation rule specifies that the range of acceptable values is ≥ 75 AND ≤ 100 , then a value of 100 is at the upper limit and a value of 75 at the lower.

Erroneous data – This is data that should not be accepted by the program. For example, a value of 0 should not be accepted by either of the validation rules given above.

Working with text files

The programs you have created so far haven't required a huge amount of data entry, but imagine typing a set of test results for everyone in your computer science class. It would take a considerable amount of time to do and – even worse – when the program terminates, all of the data will be lost. Should you need to use it again you'd have to re-enter it. This is where storing data in an external file comes in really useful. If the data you enter is stored in an external **text file** you can access it as often as you like without having to do any further keying in.

PYTHON

To access a text file, you must first give it a **file handle**, e.g. `myFile`.

Files can be opened to read from them, to write to them or to append data to them (Table 2.11).

<code>myFile = open('names.txt', 'r')</code>	Open a file named <code>names.txt</code> so that the data can be read.
<code>myFile = open('names.txt', 'w')</code>	Open a file named <code>names.txt</code> to be written to and create a new file if it does not exist. Unfortunately, if the file did exist it would overwrite all of its data.
<code>myFile = open('names.txt', 'a')</code>	Open a file named <code>names.txt</code> to add data to it.

▲ Table 2.11 Using text files

After a file has been opened it must be closed. The data is not written to a file until this command is executed.

The following program would create a text file called 'names' and add two items of data.

```
myFile = open('names.txt', 'w')
myFile.write('First item')
myFile.write('Second item')
myFile.close()
```

SUBJECT VOCABULARY

text file a sequence of lines, each of which consists of a sequence of characters

SUBJECT VOCABULARY

file handle a label that is assigned to a resource needed by the program. It can only access the file through the computer's operating system

overwritten if a file exists on the computer and a new file is created with the same name, the new file is kept and the old file is written over and lost

The file it creates would be: 'First itemSecond item'.

There is nothing to separate the two items of data. Therefore, it is useful to add a character such as a ',' or a ';' between them.

```
myFile = open('names.txt', 'w')
myFile.write('First item' + ',')
myFile.write('Second item' + ',')
myFile.close()
```

The file created would be: 'First item,Second item'.

It is important to separate the data items so that they can be read back into a data structure.

The following program would write the contents of an array into a text file called cars.txt with a comma between them.

```
Alist = ['BMW', 'Toyota', 'Audi', 'Renault', 'Rover']

myFile = open('cars.txt', 'w')

for index in range(len(Alist)):
    myFile.write(Alist[index] + ',')

myFile.close()
```

The file can be read back into an array in the following way.

```
Blist = []
myFile = open('cars.txt', 'r')

Blist = myFile.read().split(',')

MyFile.close()

Blist = myFile.read().split(',') reads the data and splits it into
separate items where there is a ','.
```

JAVA

Java programs that read and write to files often start with:

```
import system.io.*
```

This allows your Java code to access files such as File and FileReader, FileWriter from the system.io namespace.

```
import java.io.*;
class Main {
    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter("names.txt");
            fw.write("First item");
            fw.write("Second item");
            fw.close();
        }
    }
}
```

```

    } catch (IOException e) {
        System.out.println("Could not write to file");
    }
}
}
}

```

This example will write two strings to the file `names.txt`, which would then contain: `First itemSecond item`.

It is useful to separate each data value in a file with a comma (,) or new line (\n) so that they can then be read back into an array or opened as a comma separated value (CSV) file to be edited in a spreadsheet program:

```

import java.io.*;
class Main {
    public static void main(String[] args) {
        try {
            FileWriter fw = new FileWriter("names.txt");
            fw.write("First item,");
            fw.write("Second item\n");
            fw.close();
        } catch (IOException e) {
            System.out.println("Could not write to file");
        }
    }
}

```

Note that the two examples above contain `try...catch` blocks of code. These are necessary in Java because file input or output can often cause runtime errors and Java needs to know what to do instead of crashing the program.

The following program will write the contents of an array of car brands into a file called `cars.txt` with each value separated by a comma:

```

import java.io.*;
class Main {
    public static void main(String[] args) {
        String[] cars = {"BMW", "Toyota", "Audi", "Renault",
            "Rover"};
        try {
            FileWriter fw = new FileWriter("cars.txt");
            for(int index = 0; index < cars.length; index++) {
                fw.write(cars[index] + ",");
            }
            fw.close();
        } catch (IOException e) {
            System.out.println("Could not write to file");
        }
    }
}

```

The file can be read back into an array in the following way:

```
import java.io.*;
import java.util.*;
class Main {
    public static void main(String[] args) {
        String[] cars;
        try {
            File file = new File("cars.txt");
            Scanner scanner = new Scanner(file);
            String line = scanner.nextLine();

            // split each line into an array of strings
            cars = line.split(",");

            // display each car brand
            for(int index = 0; index < cars.length; index++) {
                System.out.println(cars[index]);
            }

            scanner.close();
        } catch (IOException e) {
            System.out.println("Could not read from file");
        }
    }
}
```

C#

In C#, you're likely to use the File class for reading and writing files. To use the File class you will need to add the line `Using System.IO;` at the top of your C# program, usually under the existing `Using System;` line. The File class provides over 50 different methods for working with files, so this section only provides an introduction to some of the basic methods.

WRITING TO A FILE

C#

The following program creates a file called `"names.txt"` and adds two items of data.

```
StreamWriter sw = new StreamWriter("names.txt");
sw.Write("First Item");
sw.Write("Second Item");
sw.Close();
```

The file it creates would contain: `First itemSecond item`. There is nothing to separate the two items of data. Therefore, it is useful to add a character such as a comma or semicolon between them. The file created would then contain: `First item,Second item`. It is important to separate the data items so that they can be read back into a data structure.

The following program would write the contents of an array into a text file called `cars.csv` with a comma between each item.

```
string[] makes = {"BMW", "Toyota", "Audi", "Renault", "Rover"};
StreamWriter sw = new StreamWriter("cars.csv");

foreach (string currentItem in makes)
{
    sw.Write(currentItem + ",");
}

sw.Close();
```

The file can be read back into an array using:

```
string[] makes = File.ReadAllText("z:cars.txt").Split(',');
```

PYTHON

A two-dimensional array can be stored in a text file in a similar way.

A teacher stores the names of students and their test scores in a two-dimensional array.

```
Slist = [['Faruq', 60], ['Jalila', 90]]
```

The names and scores can be saved in a text file:

```
Slist = [['Faruq', 60], ['Jalila', 90]]
myFile = open('results.txt', 'w')
```

```
for x in range(len(Slist)):
    myFile.write(Slist[x][0] + ',')
    new = str(Slist[x][1])
    myFile.write(new + '\n')
```

```
myFile.close()
```

It is easier to copy the items from a text file into a two-dimensional array in two stages.

First, copy them into a one-dimensional array:

```
Blist = []
myFile = open("results.txt", "r")
Blist = myFile.read().split(',')
myFile.close()
```

Second, move them into a two-dimensional array:

```
newList = []
for index in range(0, len(Blist) - 1, 2):
    newList.append([Blist[index], Blist[index + 1]])
```

A step of 2 is used as each record in the two-dimensional array contains 2 items from the one-dimensional array.

Chapter 10 – Subprograms

Function

In high-level programming languages, this would be written as:

PYTHON

```
def dice()
    simDie = random.randint(1, 7)
    return simDie
```

It would be called from the main program by:

```
dieThrow = dice()
```

JAVA

```
import java.util.*;
class Main {
    // define the function
    public static int dice() {
        Random r = new Random();
        return 1 + r.nextInt(6);
    }

    public static void main(String[] args) {
        // call the function
        int dieThrow = dice();
        System.out.println(dieThrow);
    }
}
```

C#

As C# is an **OO language** it has methods rather than functions. There are a number of ways of implementing methods, but it could be written like this:

```
public static int dice()
{
    Random random = new Random();
    return random.Next(1, 7);
}
```

It would be called from the main program by: `int value = dice();`

SUBJECT VOCABULARY

OO language an object-oriented programming language. Instead of data structures and separate program structures, both data and program elements are combined into one structure called an object

LOCAL AND GLOBAL VARIABLES

SUBJECT VOCABULARY

local variable a variable that is accessed only from within the subprogram in which it is created

global variable a variable that can be accessed from anywhere in the program, including inside subprograms

Notice that there are two variables that store the random number generated by the function. In the function itself, the variable `simDie` is used. This variable only exists within the function and is referred to as a **local variable**.

In the main program the value returned by the function is stored in the variable `dieThrow`. It can be used anywhere within the main program and is therefore referred to as a **global variable**.

PROCEDURES

Unlike a function, a procedure does not return a value to the main program.

ARGUMENTS AND PARAMETERS

SUBJECT VOCABULARY

parameter the names of the variables that are used in the subroutine to store the data passed from the main program as arguments

Data for the functions and procedures to work on can be passed from the main program as arguments. The function accepts them as **parameters**.

PYTHON

```
# Function rectangle
def rectangle(length, width):
    area = length * width
    return area

#Main program
rectangleLength = int(input('Please enter the length of
the rectangle'))
rectangleWidth = int(input('Please enter the width of the
rectangle'))
rectangleArea = rectangle(rectangleLength, rectangleWidth)
print(rectangleArea)
```

JAVA

```
import java.util.*;
class Main {
    // define the rectangle function
    public static int rectangle(int length, int width) {
        int area = length * width;
        return area;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Please enter the length of the
rectangle: ");
        int rectangleLength = scanner.nextInt();

        System.out.print("Please enter the width of the rectangle: ");
        int rectangleWidth = scanner.nextInt();
        scanner.close();

        int rectangleArea = rectangle(rectangleLength,
rectangleWidth);
        System.out.println(rectangleArea);
    }
}
```

C#

```
public static void Main()
{
    string lengthString, widthString;
    int length, width, rectangleArea;
```

```

Console.WriteLine("Please enter the length of the rectangle");
lengthString = Console.ReadLine();
length = int.Parse(lengthString);

Console.WriteLine("Please enter the width of the rectangle");
widthString = Console.ReadLine();
width = int.Parse(widthString);

rectangleArea = rectangle(length, width);
Console.WriteLine(rectangleArea);
}

public static int rectangle(int length, int width)
{
    int area;
    area = length * width;
    return area;
}

```

In this example, two data items are passed to the function – `rectangleLength` and `rectangleWidth`. These are called arguments.

The function receives them as parameters called `length` and `width` when the function is declared.

In the function, a variable is declared – `area`. This is called a local variable and its **scope** is within the function. If you tried to use it in the main program you would get an error message.

Lots of arguments can be passed to the function and many values can be returned.

In the following example, two values are requested and returned.

PYTHON

```

# Function rectangle
def rectangle(length, width):
    area = length * width
    circumference = (2 * length) + (2 * width)
    return area, circumference

#Main program
rectangleLength = int(input('Please enter the length of
the rectangle'))
rectangleWidth = int(input('Please enter the width of the
rectangle'))
rectangleArea, rectangleCircumference = rectangle
(rectangleLength, rectangleWidth)
print(rectangleArea)
print(rectangleCircumference)

```

SUBJECT VOCABULARY

scope the region of code within which a variable is visible

JAVA

In Java, functions can only return one value so if you need to return more than one value, you need to put them in a list or array.

```
import java.util.*;

class Main {

    // returns an array of the area and perimeter of a rectangle
    public static int[] rectangle(int length, int width) {
        int area = length * width;
        int perimeter = 2 * (length + width);
        return new int[]{area, perimeter};
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Please enter the length of the
rectangle: ");
        int rectangleLength = scanner.nextInt();

        System.out.print("Please enter the width of the
rectangle: ");
        int rectangleWidth = scanner.nextInt();
        scanner.close();

        int[] results = rectangle(rectangleLength,
rectangleWidth);
        int rectangleArea = results[0];
        int rectanglePerimeter = results[1];
        System.out.println("Area: " + rectangleArea);
        System.out.println("Perimeter: " + rectanglePerimeter);
    }
}
```

C#

Methods in C# can't return multiple values unless you use **By Reference**, which is beyond the scope of this book.

Subprograms and Menues

Functions and procedures are useful when using menus in a program. When a user selects a menu option, they can be sent to a particular function or procedure.

The benefits of using subprograms

Repeated sections of code need only be written once and called when necessary. This shortens the development time of a program and means that the finished program will occupy less memory space when it is run.

If changes have to be made at a later date it is easier to change a small module than having to work through the whole program. In large development teams different members can be working independently on different subprograms. They can use and develop standard libraries of subroutines that can be reused in other programs.

Built in functions

In addition to user-written subprograms, most high-level programming languages have a set of built-in functions for common tasks. These are designed to save the programmer time, such as functions that print, count the number of characters in a string and generate random numbers.

SUBJECT VOCABULARY

built-in functions functions that are provided in most high-level programming languages to perform common tasks

Chapter 11 – Testing and evaluation

Logic errors

Logic errors occur when the thinking behind an algorithm is incorrect so that the output isn't what is expected or intended. Ideally, logic errors should be identified and fixed at the design stage. The following algorithm is intended to work out whether a learner has passed a test. Learners need a score of 80 or above to pass. However, a logic error in the algorithm means that it produces an incorrect and unexpected result.

```
IF testScore <= 80 THEN
    SEND 'Pass' TO DISPLAY
ELSE
    SEND 'Failed' TO DISPLAY
END IF
```

SUBJECT VOCABULARY

logic error an error in an algorithm that results in incorrect or unexpected behaviour

trace table a technique used to identify any logic errors in algorithms. Each column represents a variable or output and each row a value of that variable

TRACE TABLES

The formal way of checking the logic of an algorithm is to use a trace table.

Syntax errors

Syntax errors occur when the grammar rules of a programming language are not followed. They prevent the code from being compiled or translated. Examples of syntax errors are:

- writing 'prnit' instead of 'print'
- missing out a closing bracket
- missing out quotation marks

RUNTIME ERRORS

SUBJECT VOCABULARY

runtime error an error that occurs while the program is running – the operation the computer is asked to do is impossible to execute

Runtime errors occur during program execution and are the most difficult to predict and spot.

This program is designed to take two numbers, divide the first number by the second number and output the result. It will work as intended at least some of the time. However, if the user entered 5 and 0, a runtime error would occur because it is impossible for the computer to divide 5 by 0.

Python

```
firstNumber = int(input('Please enter the first number'))
secondNumber = int(input('Please enter the second number'))
result = firstNumber / secondNumber
print(result)
```

Java

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Please enter the first number: ");
        int firstNumber = scanner.nextInt();
        System.out.print("Please enter the second number: ");
        int secondNumber = scanner.nextInt();
        scanner.close();

        // do the division (converting all integers to reals)
        double result = (double)firstNumber / (double)
secondNumber;
        System.out.println(result);
    }
}
```

C#

```
int firstNumber, secondNumber, result;
string firstNumberString, secondNumberString;

Console.WriteLine("Please enter the first number");
firstNumberString = Console.ReadLine();
firstNumber = int.Parse(firstNumberString);

Console.WriteLine("Please enter the second number");
secondNumberString = Console.ReadLine();
secondNumber = int.Parse(secondNumberString);

result = firstNumber / secondNumber;
Console.WriteLine(result);
```

ERROR SUMMARY

This table gives a summary of the three types of error you are likely to encounter.

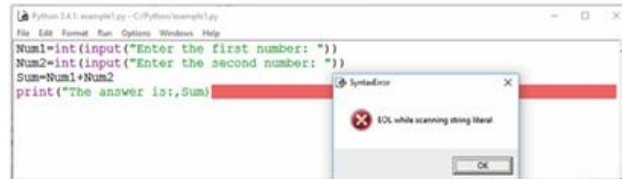
TYPE OF ERROR	DESCRIPTION
Logic	The program seems to run normally; however, there is an error in the logic of the program, which means it does not produce the result you expect.
Syntax	Syntax refers to the rules of the programming language. A syntax error means that part of the code breaks the rules of the language, which stops it running.
Runtime	An error that occurs when the computer tries to run code that it cannot execute.

▲ Table 2.14 Summary of the three types of error you are likely to encounter

Using an integrated development environment (IDE)

You probably already have first-hand experience of using an Integrated Development Environment (IDE) when writing code. It's definitely worth taking some time to get to know the IDE that comes with the language you are using.

► Figure 2.5 A syntax error flagged up by an IDE in Python code



SUBJECT VOCABULARY

Integrated Development Environment (IDE) a package that helps programmers to develop program code. It has a number of useful tools, including a source code editor and a debugger

debugger a computer program that assists in the detection and correction of errors in other computer programs

THE TEST PLAN

At the start of a programming project, it is crucial to make a list of the requirements that the program is expected to meet. Throughout the development phase of the project, you should regularly refer back to this list of requirements to check that you are on track to achieve them.

TEST NO.	PURPOSE OF THE TEST	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	ACTION NEEDED/ COMMENTS
1	To check correct conversion of valid mark	0	'FAIL'		
		55	'D'		
		65	'C'		
		75	'B'		
		85	'A'		
2	To check correct conversion of boundary mark	0	'FAIL'		
		1, 59	'D'		
		60, 69	'C'		
		70, 79	'B'		
		80, 100	'A'		
3	To check response to erroneous mark	-5	Error message		
		105	Error message		

▲ Table 2.15 Test plan extract

Normal data	Data that is well within the limits of what should be accepted by the program.	Test 1 uses normal data to check if marks are converted into grades correctly (e.g. a mark of 65 should be converted to a grade C).
Boundary data	Data that is at the outer limits of what should be accepted by the program.	Test 2 uses boundary data to check that the program works correctly with marks at the upper and lower boundaries (e.g. a mark of 60 and a mark of 69 should both convert to a grade C).
Erroneous data	Data that should not be accepted by the program.	Test 3 uses erroneous data to check that the program does not accept it.

▲ Table 2.16 Test data categories

Testing is just as important as writing code because it ensures that the finished program works correctly and fully meets the requirements. You should use a 'bottom up' approach to testing (i.e. test each subprogram as you develop it and then test the whole program once it is finished). Here is the completed test plan for the grade calculator program.

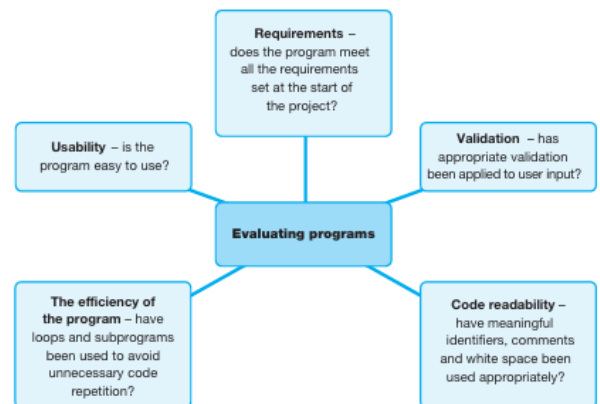
TEST NO.	PURPOSE OF THE TEST	TEST DATA	EXPECTED RESULT	ACTUAL RESULT	ACTION NEEDED/ COMMENTS
1	To check correct conversion of valid mark	0 55 65 75 85	'FAIL' 'D' 'C' 'B' 'A'	'FAIL' 'D' 'C' 'B' 'A'	None
2	To check correct conversion of boundary mark	0 1, 59 60, 69 70, 79 80, 100	'FAIL' 'D' 'C' 'B' 'A'	'FAIL' 'D' 'C' 'B' 'A'	None
3	To check response to erroneous mark	-5 105	Error message Error message	'FAIL' 'A'	A range check has been added to ensure that only valid marks can be added.

▲ Table 2.17 The completed test plan for the grade calculator program

Evaluating programs

You need to be able to identify the strengths and weaknesses of your own programs as well as those created by other programmers. This will enable you to identify techniques that work well and aspects that could be improved.

► Figure 2.6 Consider the aspects shown when evaluating a program



Revision questions

(2023 June)

1. Programs are used to handle financial transactions.
 - a. Monthly account statements are created by a program.

Figure 1 shows a statement.

Date	Description	Debit (£)	Credit (£)	Balance (£)
01 Jan	Opening balance			128.35
04 Jan	Music store	10.23		118.12
07 Jan	Salary		1,515.28	1,633.40
11 Jan	Electricity company	50.00		1,583.40
19 Jan	Uniform shop	12.56		1,570.84
27 Jan	Refund		25.00	1,595.84
29 Jan	Rent	750.00		845.84
31 Jan	Interest		0.25	846.09

Figure 1

Complete the table to identify an input, an output and a process used by the program to generate the **furthest right column**.

(3)

Input	
Processing	
Output	

(b) A tax rate is applied to a gross value to give a net value.

Open Q01b in the code editor.

Use the code to answer these questions.

- (i) Give the contents of a comment. (1)
- (ii) Identify a logical operator used in this program. (1)
- (iii) Give the name of a global variable. (1)
- (iv) Give the name of a constant. (1)
- (v) Give the name of a parameter. (1)
- (c) Variables and constants are used in program code.
 - (i) State the purpose of a constant. (1)
 - (ii) Give one benefit of using a constant. (1) (Total 10)

2). Solutions to problems are made up of many different components.

(a) Programmers use subprograms when developing code.

(i) Give two reasons to use subprogram libraries when developing code. (2)

(ii) Identify which one of these must return a result. (1)

- ☐ A Function
- ☐ B Iteration
- ☐ C Procedure
- ☐ D Selection

(iii) State what is meant by the term local variable. (1)

(b) A school is planning a fish and chip dinner for the students and their families.

(i) Tickets are only sold to families.

A family is one to four adults and one to six children.

A program is being written to help manage ticket sales.

Complete the table to show examples of normal, boundary and erroneous numeric test data for the program. (3)

	Adult	Children
Normal		
Boundary		
Erroneous		

ii) The kitchen staff use a program to determine whether they have enough chips or how many more they need to order.

Open Q02bii in the code editor.

There are four errors in the code.

Amend the code to correct the errors.

Use this test data to help you find the errors.

(3)

Chips in stock (kilograms)	Number of adults	Number of children	Expected output
19	100	150	Stocks available
15	100	150	Order 4.0 kilograms

Save your amended code as Q02biiFINISHED with the correct file extension for the programming language.
(11 Marks)

3). Algorithms can be represented in flowcharts, pseudocode or program code.

(a) Trace tables can be used with flowcharts or pseudocode. Give two characteristics of a trace table.

(b) An algorithm has been written to validate numbers entered by the user.

Figure 2 shows the pseudocode for the algorithm.

```

1 RECEIVE num1 FROM (INTEGER) KEYBOARD RECEIVE
2 num2 FROM (INTEGER) KEYBOARD
3 IF ((num1 < 16) AND (num2 < 23)) OR (num2 = 13)) THEN SEND
4     "State 1" TO DISPLAY
5 ELSE
6     IF ((num2 > 12) AND (num1 > 20)) THEN
7         SEND "State 2" TO DISPLAY
8     ELSE
9         IF ((num1 = 88) OR NOT (num2 = 18)) THEN
10            SEND "State 3" TO DISPLAY
11        ELSE
12            SEND "State 4" TO DISPLAY END
13        IF
14            END IF END
15 IF

```

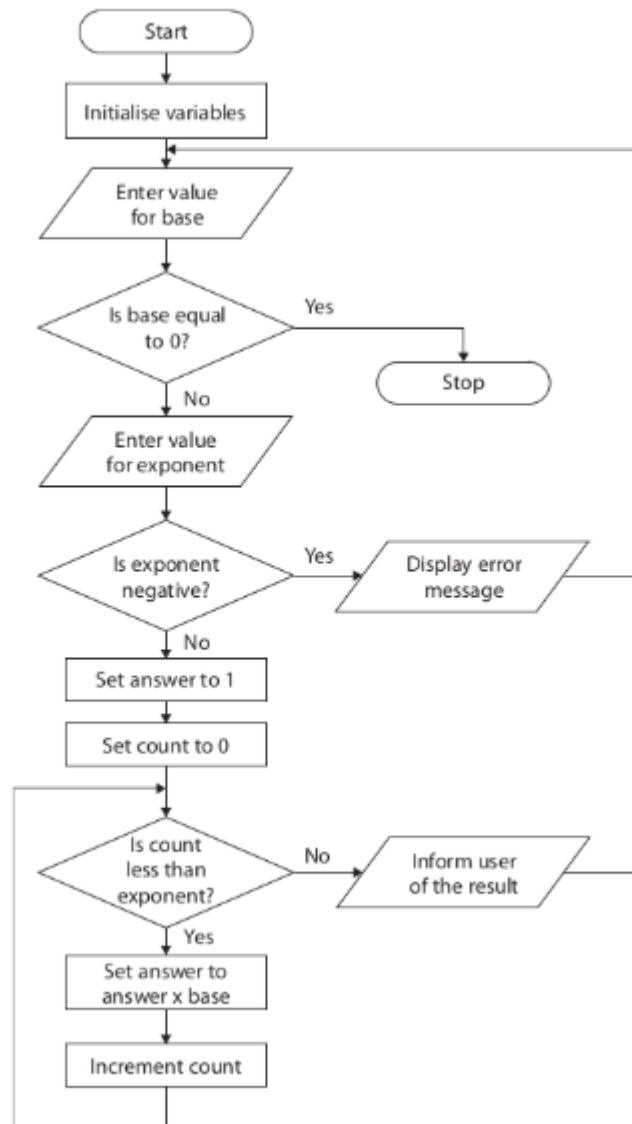
Figure 2

Complete the table to show the output for each set of inputs.

num1	num2	Output
88	18	
17	18	
12	19	

(c) A program is required to calculate the result of raising one integer (the base) to the power of another (the exponent).

Figure 3 shows a flowchart for the algorithm.



The program has these requirements:

- outputs meaningful error messages
- outputs the final answer with the base and the exponent.

Open Q03c in the code editor.

Write a program to implement the logic in the flowchart.

Do not add any further functionality.

Save your code as Q03cFINISHED with the correct file extension for the programming language. (10)

(Total for Question 3 = 15 marks)

4. A program is needed to create a key for a database.

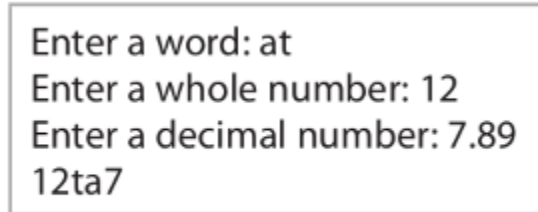
A user enters a two-letter word, a whole number and a decimal number.

The program must ensure the word is only two characters long.

The program must display an error message when the word is not two characters long.

A key is generated from the whole number, the reversed word and the whole number part of the decimal number.

Figure 4 shows the input values and a valid key.



```
Enter a word: at
Enter a whole number: 12
Enter a decimal number: 7.89
12ta7
```

Figure 4

Open Q04b in the code editor.

Write the program.

Do not add any further functionality.

Save your code as Q04bFINISHED with the correct file extension for the programming language. (8)

5. A marine scientist is conducting a study of tuna in the world's oceans.

(a) A list of tuna species needs to be sorted into descending alphabetical order using a merge sort algorithm.

Figure 5 shows the lists created at the end of the splitting process.



Bigeye	Blackfin	Albacore	Longtail	Bluefin
--------	----------	----------	----------	---------

Figure 5

Each list is a single tuna species.

It will require three passes to merge the lists into a single sorted list in descending order.

Complete the merge sort using the space provided. (8)

(b) A bubble sort could be used to sort the list of tuna species into ascending order.

Figure 6 shows an algorithm for a bubble sort.

```

1  SET myTuna TO ["Bigeye", "Blackfin", "Albacore",
2                  "Longtail", "Bluefin"]
3  SET tmp TO 0
4  SET swaps TO True
5    SET length TO LENGTH
   (myTuna) 6
7  WHILE (swaps = True) DO
8    SET swaps TO False
9    FOR ndx FROM 0 TO length - 1 DO
10     IF myTuna[ndx] > myTuna[ndx + 1] THEN
11       SET tmp TO myTuna[ndx + 1]
12       SET myTuna[ndx] TO myTuna[ndx + 1]
13       SET myTuna[ndx + 1] TO tmp
14       SET swaps TO True
15     END IF
16   END FOR
17 END WHILE

```

Figure 6

There is an error in the loop between line 9 and line 16.

Complete the table to give the line number with an error and a corrected line of pseudocode. (2)

Line number with error	
Corrected line of pseudocode	

c. The scientist is collecting and storing data about tuna.

The data collected is:

- species
- length in centimetres
- weight in kilograms
- age in years.

The data is stored in an array.

The collected data needs to be written to a file named TunaData.txt

Each record stored in the file must have a code number in the first field.

Code numbers must start at 101

Each field in a record should be separated by a comma.

Figure 7 shows the contents of the file.

```
101, Yellowfin, 105, 15, 3
102, Albacore, 90, 15, 5
103, Skipjack, 50, 3, 4
104, Bigeye, 105, 25, 4
105, Atlantic Bonito, 50, 4, 2
106, Northern Bluefin, 190, 120, 11
107, Southern Bluefin, 190, 120, 11
108, Tongol, 90, 20, 4
```

Figure 7

Open ****Q05c**** in the code editor.

Write the program.

You must use the structure given in ****Q05c**** to write the program.

Do not add any further functionality.

Save your code as ****Q05cFINISHED**** with the correct file extension for the programming language. (6)

(Total for Question 5 = 10 marks)

6. An agricultural college has a herd of dairy cows.

Data collected for the cows is stored in arrays.

The data stored is:

- the name of the breed
- the ease of care rating for that breed (1 is highest and 3 is lowest)
- the number of cows in the herd of that breed
- the volume of milk per day for a cow of that breed.

The college wants to present the data to farmers and to recommend the best breed of cows for them.

The best breed has the highest care rating and the largest volume of milk per day for a cow of that breed.

Open the file Q06 in the code editor.

Write a program to:

- calculate the daily volume of milk produced by each breed
- add this daily volume to the data structure `tbl_dailyVolume`
- display a message informing the user what each field holds, such as breed, rating, volume per cow, count and total volume
- display the data for each breed
- calculate and display the total volume of milk produced each day by the herd
- find the recommended breed
- display the recommended breed by name.

Figure 8 shows the output from a functional program.

```
Fields are: Breed, Rating, Volume (cow), Count, Volume (day)
Red Chittagong 1 7.5 6 45.0
Sussex 2 5.7 3 17.1
Dexter 3 11.4 8 91.2
Abondance 2 11.4 7 79.8
Sahiwal 3 22.0 6 132.0
Vorderwald 1 15.2 4 60.8
Ayrshire 2 21.0 3 63.0
Jersey 1 18.3 7 128.1
Randall 2 19.0 3 57.0
Alderney 1 9.0 3 27.0
Carora 3 23.1 4 92.4
Gloucester 2 16.0 7 112.0
Total: 905.4 litres
Recommended breed: Jersey rating: 1 volume 18.3
```

Your program should function correctly even if the number of breeds represented in the data is changed.

Save your amended code as Q06FINISHED with the correct file extension for the programming language.

2022 June

1). Programmers write code to solve problems.

(a) Identify the description of a variable in a computer program. (1)

- A A value that cannot be used more than once
- B A value that must be input
- C A value that is always used in a calculation
- D A value that can change

(b) Identify the technique that improves the readability of code. (1)

- A Using indents on every line
- B Using descriptive names for variables
- C Using the correct operators
- D Using suitable data structures

(c) Complete the table by giving an example value for each data type.

The first row has been completed for you. (2)

Data type	Example value
integer	12
char	
real	

(d) Describe one difference between the data used in boundary testing and the data used in erroneous testing. (2)

(e) A program should output the value 2

However, there is an error in the code and the actual output is 4

Name this type of error. (1)

(f) Open Q01f in the code editor.

The program should allow the input of the length of the side of a square and output the area of the square.

There are three errors in the code.

Amend the code to correct the errors.

Save your amended code as Q01fFINISHED with the correct file extension for the programming language. (3)

(g) A program is needed that will accept the input of a letter and compare it with a stored letter.

It will check whether the letter comes earlier in the alphabet, later in the alphabet or is the same letter as the stored letter.

It will output the letter and the stored letter with an appropriate message.

Open Q01g in the code editor.

Amend the code to complete the if statement used to produce the output.

You must use the structure given in Q01g to complete the program.

Do not add any further functionality.

Save your code as Q01gFINISHED with the correct file extension for the programming language. (4)

(Total for Question 1 = 14 marks)

2). Raza is writing a program to tell the user whether a number they input is a prime number.

A prime number is a whole number, larger than one, that can only be divided by one and itself with no remainder.

This pseudocode contains the logic required to complete the program.

```
1 FUNCTION checkPrime(pNumber)
2 BEGIN FUNCTION
3     IF pNumber = 1 THEN
4         check = False
5     ELSE
6         check = True
7         FOR count FROM 2 TO pNumber DO
8             IF pNumber MOD count = 0 THEN
9                 check = False
10            END IF
11        END FOR
12    END IF
13 RETURN check
14 END FUNCTION
15
16 SEND "Enter a number: " TO DISPLAY
17 RECEIVE number FROM (INTEGER) KEYBOARD
18 SET result TO checkPrime(number)
19
20 IF result = True THEN
21     SEND (number & " is a prime number") TO DISPLAY
22 ELSE
23     SEND (number & " is not a prime number") TO DISPLAY
24 END IF
```

(a) Answer these questions about the pseudocode.

(i) Identify a line number where selection starts. (1)

(ii) Identify the line number where iteration starts. (1)

(iii) Identify a line number where a string and a number are printed on the same line. (1)

(iv) Identify the name of a local variable. (1)

(v) Identify the name of a parameter. (1)

(b) Write a program to implement the logic in the pseudocode.

Open Q02b in the code editor.

Write the program.

You must use the structure given in Q02b to complete the program.

Do not add any further functionality.

Save your code as Q02bFINISHED with the correct file extension for the programming language. (11)

(Total for Question 2 = 16 marks)

3). Manjit sells copies of a science textbook to schools.

She needs a program to process textbook orders. It must:

- accept the input of the number of textbooks required
- generate the price per textbook
- generate the total cost of the order
- display the price per textbook and the total cost of the order.

The price of one textbook depends on the number ordered:

Quantity	Price per textbook
1-5	£20.00
6-9	£15.00
10 or more	£12.00

Open Q03 in the code editor.

Write the program.

You must use the structure given in Q03 to complete the program.

Do not add any further functionality.

Save your code as Q03FINISHED with the correct file extension for the programming language.

(Total for Question 3 = 6 marks)

4). Julia runs a computer gaming club.

(a) She wants a program to check passwords stored in a file.

The file passwords.txt contains the list of passwords.

The program must:

- check each password to ensure that:
- the first character is an uppercase letter
- if it is, that it also includes at least one digit (0-9)
- if a password does not meet these requirements:

- display the password
- increment the number of incorrect passwords
- display the total number of incorrect passwords after all the passwords have been checked.

Open Q05a in the code editor.

Write the program.

You must use the structure given in Q05a to write the program.

Do not add any further functionality.

Save your code as Q05aFINISHED with the correct file extension for the programming language. (9)

(b) Figure 3 shows an array that stores player scores after a game.

3	2	10	8	1	9
---	---	----	---	---	---

Figure 3

(i) Julia uses a bubble sort algorithm to sort the scores.

Complete the table to show how the bubble sort algorithm will sort the scores.

You may not need to use all the rows.

3	2	10	8	1	9

(ii) Explain one reason why a bubble sort is very efficient in terms of memory usage. (2)

(Total for Question 5 = 17 marks)

5).

Carlos wants you to create a guess the animal game.

Open Q06 in the code editor.

The code contains an array of animals.

It also contains a function that randomly selects an animal from the array. This is the secret word the user needs to guess.

Carlos wants the program to:

- generate the number of attempts the user has to guess the secret word.

The maximum number of attempts is the length of the secret word +3. For example, the user has 8 attempts to guess when the secret word is tiger

- keep track of letters from incorrect attempts that are in the secret word and those that are not. There should be no duplicated letters
- display a message telling the user:
- the number of letters in the secret word
- how many attempts they have left
- force the user to input a word that is the same length as the secret word
- check whether the input word matches the secret word:
- if the words match then a message that includes the secret word and the number of attempts taken to guess it is displayed
- if the words do not match then:
- letters from the attempt that appear in the secret word should be added to the correct letters store
- letters from the attempt that do not appear in the secret word should be added to the wrong letters store
- the contents of the correct and wrong letter stores are displayed
- allow the user another attempt until they have guessed the word or have run out of attempts
- display a message telling the user the game is over including the random word if the maximum attempts have been taken and the word has not been guessed.

Figure 4 shows the contents of the correct and wrong stores after two attempts to guess the secret word cow.

Secret word cow			
First attempt		Second attempt	
input	dog	input	cat
correct store	o	correct store	o c
wrong store	d g	wrong store	d g a t

Figure 4

Your program should include at least two subprograms that you have written yourself.
 You must include comments in the code to explain the logic of your solution.
 Save your code as Q06FINISHED with the correct file extension for the programming language.
 (Total for Question 6 = 20 marks)