Edexcel

OL IGCSE

Computer science

CODE: (4CP0)

Unit 01

Problem solving

1.1 Understanding algorithms

An example of an algorithm

An interactive map is a useful way to find a route between two locations. Figure 1.1 shows a route between two cities that was calculated by a mapping program.

The route on this interactive map has been calculated using an algorithm.

It is **unambiguous** in telling the driver exactly what to do, like 'turn left', 'turn right' or 'go straight'.
It is a **sequence** of steps.
It can be used again and will always provide the same result.
It provides a solution to a problem, in this case, how to get from Beijing to Shanghai.

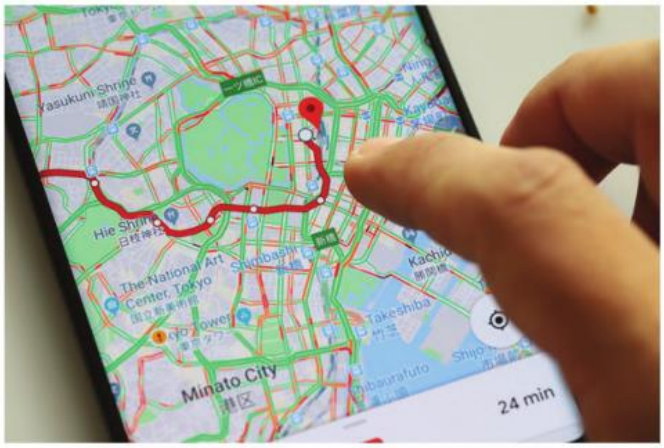
A solution to a problem with these characteristics is called an **algorithm**. Most problems have more than one solution, so different algorithms can be created for the same problem.

SUBJECT VOCABULARY

unambiguous this means that the instructions cannot be misunderstood. Simply saying 'turn' would be ambiguous (i.e. unclear) because you could turn left or right. All instructions given to a computer must be unambiguous or it won't know what to do

sequence an ordered set of instructions

algorithm a precise method for solving a problem



Successful algorithm

There are three points to consider when deciding whether an algorithm is successful or not.

Accuracy - it must lead to the expected **outcome** (e.g. create a route from Beijing to Shanghai).
Consistency - it must produce the same result each time it is run.
Efficiency - it must solve the problem in the shortest possible time, using as few computer resources as possible.

The relationship between algorithms and programs

Algorithms and programs are closely related, but they are not the same. An algorithm is a detailed design for a solution; a program is when that design is implemented.

SUBJECT VOCABULARY

high-level programming language a programming language that is similar to natural human language

This unit is all about algorithms. We look at how algorithms are implemented in **high-level programming languages** in Unit 2.

Displaying and algorithms

An algorithm can be expressed in different ways.

WRITTEN DESCRIPTIONS

A written description is the simplest way of expressing an algorithm. Here is an algorithm describing the everyday task of making a cup of instant coffee:

ALGORITHM FOR MAKING A CUP OF COFFEE

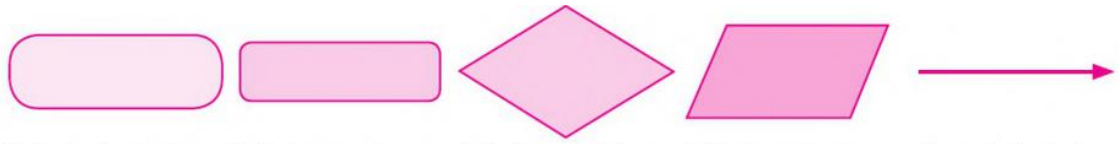
Fill kettle with water.
Turn on kettle.
Place coffee in cup.
Wait for water to boil.
Pour water into cup.
Add milk and sugar.
Stir.

FLOW CHARTS

Flowcharts can be used to show an algorithm as a diagram. They provide a more visual display.

There are special symbols that have to be used in a flowchart. You can't just make up your own, because nobody else would be able to follow your algorithm.

Figure 1.2 shows the flowchart symbols that should be used.

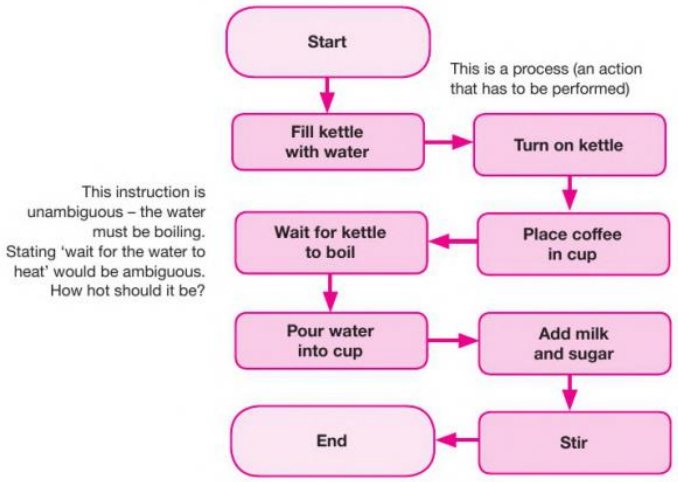


Indicates the start or end of an algorithm | Indicates a process to be carried out | Indicates a decision to be made | Indicates an input or output | Shows the logical flow of the algorithm

▲ **Figure 1.2** Flowchart symbols

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

▶ **Figure 1.3** Flowchart of an algorithm to make a cup of coffee



PSEUDOCODE

In addition to flowcharts and written descriptions, algorithms can also be expressed in **pseudocode**. The pseudocode can be used to code the solution in an actual programming language.

It allows the **developer** to concentrate on the **logic** and efficiency of the algorithm without having to bother about the rules of any programming language.

ARITHMETIC OPERATORS

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

▲ Table 1.1 Arithmetic operators

SUBJECT VOCABULARY

pseudocode a structured, code-like language that can be used to describe an algorithm

developer a person whose job it is to create new software

logic the principles and reasoning underlying the constructs and elements to be applied in solving problems

Variable and constants

A **constant** is the opposite of a variable. It is a 'container' that holds a value that always stays the same. Constants are useful for storing fixed information, such as the value of pi, the number of litres in a gallon or the number of months in a year. Each variable and constant in an algorithm have to have a unique identifier. It is important to choose descriptive names for identifiers. This will make your code much easier to read.

NAMING CONVENTIONS FOR VARIABLES AND CONSTANTS

It is sensible to write identifiers in the same way throughout an algorithm. A common method is to use 'camel case' for compound words (e.g. firstName, secondName) with no space between words and the second word starting with a capital letter. Alternatively, you could capitalise the first letter of both words, e.g. FirstName, SecondName, or separate the words with an underscore, e.g. first_name, second_name, known as 'snake case'.

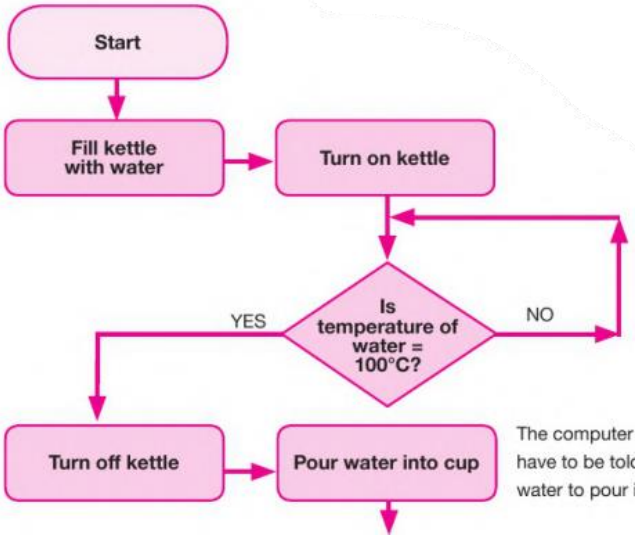
SUBJECT VOCABULARY

constant a 'container' that holds a value that never changes; like variables, constants have unique identifiers

1.2 Creating algorithms

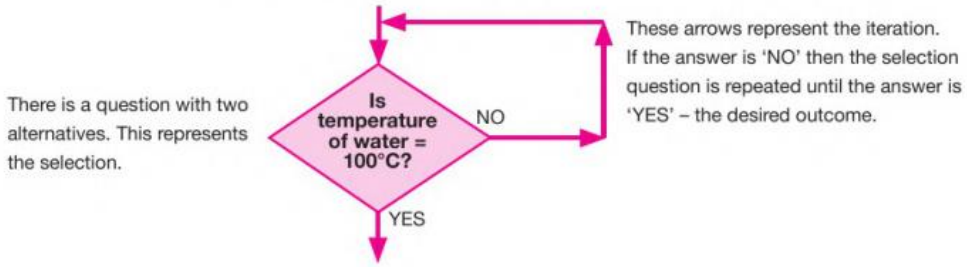
Algorithms for computers

A version of this part of the algorithm, suitable for a computer, is shown in Figure 1.6. This example introduces two new **constructs** from which algorithms are created. We have already met the construct sequence - step-by-step instructions in the correct order. To add to this, we now have **selection** and **iteration**.



▶ **Figure 1.6** Part of an algorithm suitable for a computer for making coffee

Representing selection and iteration in a flowchart



▶ **Figure 1.7** Selection and iteration in a flowchart

ITERATION

When writing programs, it is often necessary to repeat the same set of statements several times. Instead of making multiple copies of the statements, you can use iteration to repeat them. The algorithm for making a cup of coffee includes an instruction to keep waiting until the water in the kettle boils.

1.3 Sorting and searching algorithms

Sorting algorithms

BUBBLE SORT

When data is sorted, different items must be compared with each other and moved so that they are in either **ascending order** or **descending order.**

The bubble sort algorithm starts at one end of the list and compares pairs of data items. If they are in the wrong order, they are swapped. The comparison of pairs continues to the end of the list, each complete **traversal** of the list being called a 'pass'.

SUBJECT VOCABULARY

ascending order this is arranging items from smallest to largest (e.g. 1, 2, 3)

descending order this is arranging items from largest to smallest (e.g. 3, 2, 1)

traversal travel across or through something.

MERGE SORT

Merge sort is a sorting algorithm that divides a list into two smaller lists and then divides these until the size of each list is one. Repeatedly applying a method to the results of a previous application of the method is called **recursion**.

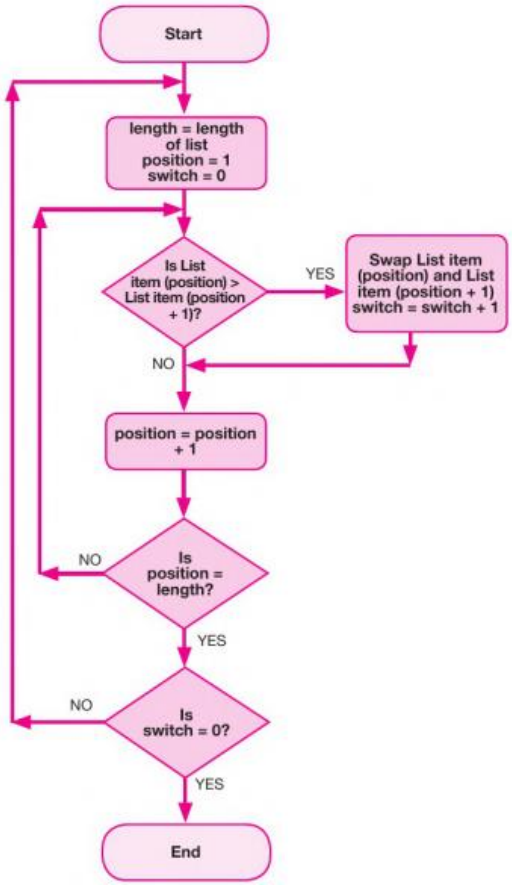
SUBJECT VOCABULARY

recursion a process that is repeated. For example, a document can be checked and edited, checked and edited and so on until it is perfect

EFFICIENCY OF SORTING ALGORITHMS

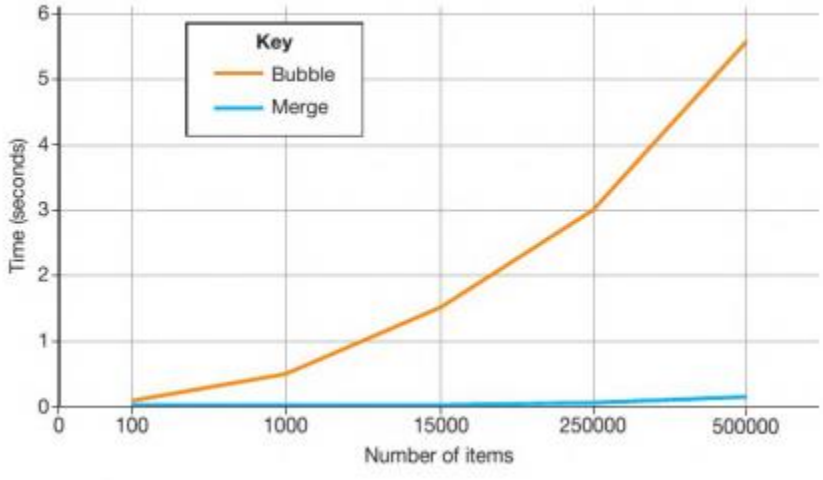
The bubble sort algorithm is said to be using **brute force** because it starts at the beginning and completes the same task repeatedly until it has found a solution.

▶ **Figure 1.9** Bubble sort algorithm written as a flowchart



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

This graph compares the performance of the bubble and merge sort algorithms.



SUBJECT VOCABULARY

brute force an algorithm design that does not include any techniques to improve performance, but instead relies on computing power to try all possibilities until the solution to a problem is found

divide and conquer an algorithm design that works by dividing a problem into smaller and smaller sub-problems, until they are easy to solve. The solutions to these are then combined to give a solution to the complete problem

SERCHING ALGORITHMS

LINEAR SEARCH

A linear search is a simple algorithm and not very sophisticated. It simply starts at the beginning of the list and goes through it, item by item, until it finds the item it is looking for or reaches the end of the list without finding it.

SUBJECT VOCABULARY

median the middle number when the numbers are put in ascending or descending order (e.g. if there are 13 numbers, then the 7th number is the median). If there are an even number of items in a list, the median is the mean of the middle two numbers (e.g. if there are 10 numbers, add the 5th and 6th numbers together and divide the result by 2). In a binary search, the higher of the two numbers would be chosen

BINARY REASERCH

Like a merge sort, a binary search uses a 'divide and conquer' method.
In a binary search the middle or **median** item in a list is repeatedly selected to reduce the size of the list to be searched - another example of recursion.

EFFICIENCY OF SEARCHING ALGORITHMS

In the example on page 20, the linear search was more efficient because it only had to carry out two comparisons instead of the three for a binary search. But is this always the case?

Searching algorithms can be compared by looking at the 'worst case' and the 'best case' for each one.

1.4 Decomposition and subtraction

Problem solving

The tasks of a computer scientist include defining and analysing problems; creating structured solutions - algorithms; and coding the solutions into a form that can be implemented by a computer. These tasks are part of what is known as **computational thinking.**

One of the skills required for computational thinking is algorithm design (which we've covered in detail in this unit). If there is a fault in the algorithm design, then the program will not work, however good a coder you are. Two other skills are **decomposition and abstraction.**

SUBJECT VOCABULARY

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

decomposition breaking a problem down into smaller, more manageable parts, which are then easier to solve

abstraction the process of removing or hiding unnecessary detail so that only the important points remain

Decomposition

Decomposition is the initial step in problem-solving, allowing teams to work on sub-problems simultaneously. This approach simplifies error detection and correction, and when developed into a program, code can be reused.

Abstraction

Absorption is a common technique in daily life, allowing us to understand the essential features of something. For example, when someone describes seeing a cat, we can recognize it as a small, furry animal with four legs and a tail, allowing us to imagine it when talking about it.

LEVELS OF ABSTRACTION

There are different levels or types of abstraction. The higher the level of abstraction, the less detail is required. We use abstraction all the time in accomplishing everyday tasks.

When programmers write the 'print' command they do not have to bother about all of the details of how this will be accomplished. They are removed from them. They are at a certain level of abstraction.

A driver turning the ignition key to start a car does not have to understand how the engine works or how the spark to ignite the petrol is generated. It just happens and they can simply drive the car. That is abstraction.

Coding an algorithm

High-level programming languages make it easier for a programmer to write code. Unfortunately, the processor that has to execute the program cannot understand the language it is written in. It therefore needs a translator to translate the code into the only language it does understand - a stream of 1s and Os.

These high-level languages are therefore at a high level of abstraction - very far removed from the actual language of a computer.

The processing can be split into parts. For example, in the example of the noughts and crosses game there could be separate algorithms for:

- Deciding where the computer should make its next selection - it could be called 'computer entry'
- Checking if the computer or the player has won - it could be called 'check if won'
- Checking if there are any empty squares left - it could be called 'check draw'.

These separate algorithms could be used when they are needed. It is efficient because it means that the same code doesn't have to be rewritten whenever it is needed.
These items of code are called **subprograms**.

Revision questions

1). (2023 may:6)
Programmers share algorithms with different people and write algorithms for
different reasons.
(a) A programmer is showing a new algorithm to a group of non-technical managers.
State an appropriate method for writing the algorithm.
Justify your answer.(2)
Method ……..
Justification ……..
(b) **Figure 1** shows an algorithm that displays a string based on the number input by
the user.

```
1  SEND ("Enter a number: ") TO DISPLAY
2  RECEIVE inNum FROM (INTEGER) KEYBOARD
3  IF ((inNum = 1) OR (inNum = 2)) THEN
4      IF (inNum = 1) THEN
5          SEND ("First") TO DISPLAY
6      ELSE
7          IF (inNum = 2) THEN
8              SEND ("Second") TO DISPLAY
9          END IF
10     END IF
11 ELSE
12     SEND ("Invalid input") TO DISPLAY
13 END IF
```

Figure 1

Give one reason why the selection statement on line 7 is not required. (1)

(c) Figure 2 shows an algorithm that manipulates arrays.

The algorithm works with any number of scores.

```
1 SET oldScores TO [10, 20, 30, 40, 50]
2 SET newScores TO [0, 0, 0, 0, 0]
3 SET newIndex TO 0
4
5 FOR oldIndex FROM (LENGTH (oldScores) - 1) TO 0 STEP -1 DO
6     SET newScores[newIndex] TO oldScores[oldIndex]
7     SET newIndex TO newIndex + 1
8 END FOR
```

Figure 2

(i) Describe what happens to the variable oldIndex when line 5 is executed. (2)

(ii) State the purpose of the algorithm in Figure 2. (1)

3). (2023 may:paper 2:3)

lgorithms 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. (2)

(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
 2 RECEIVE num2 FROM (INTEGER) KEYBOARD
 3 IF (((num1 < 16) AND (num2 < 23)) OR (num2 = 13)) THEN
 4     SEND "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
13         END IF
14     END IF
15 END IF
```

Figure 2

Complete the table to show the output for each set of inputs. (3)

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.

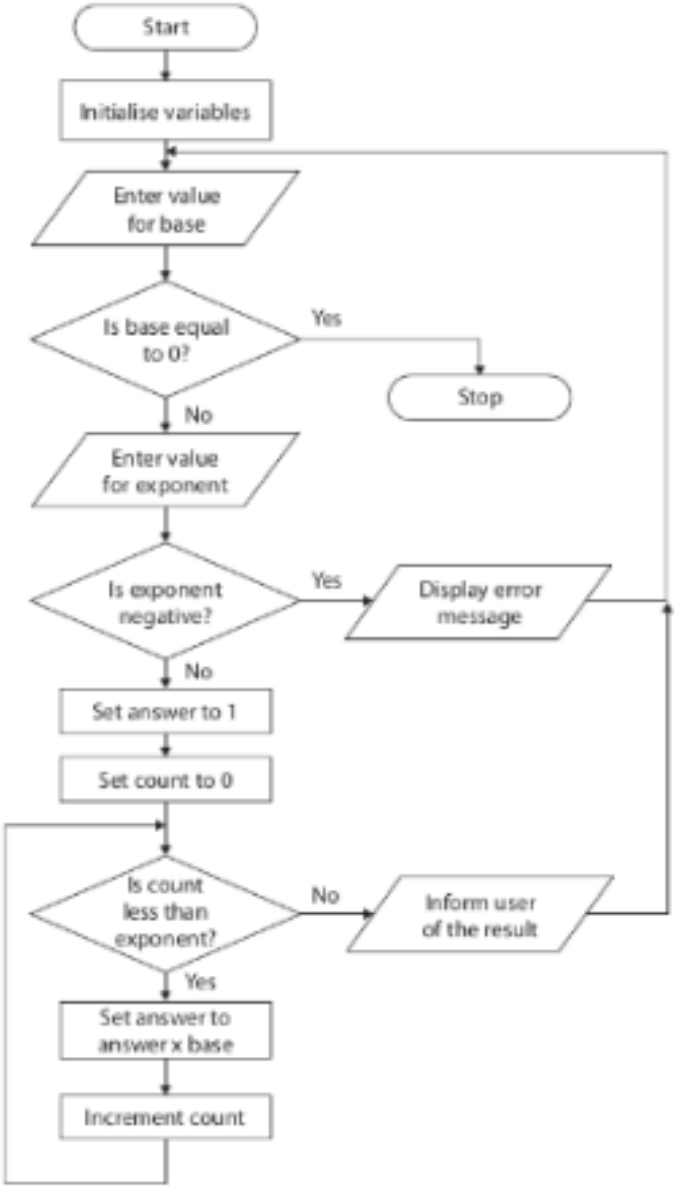


Figure 3

Explain what is happening in this program [15]

3). (2023 may:paper 2:5)
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.
Each list is a single tuna species.

Figure 5
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.
(2)

(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
```

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 pseudocod

Line number with error	
Corrected line of pseudocode	

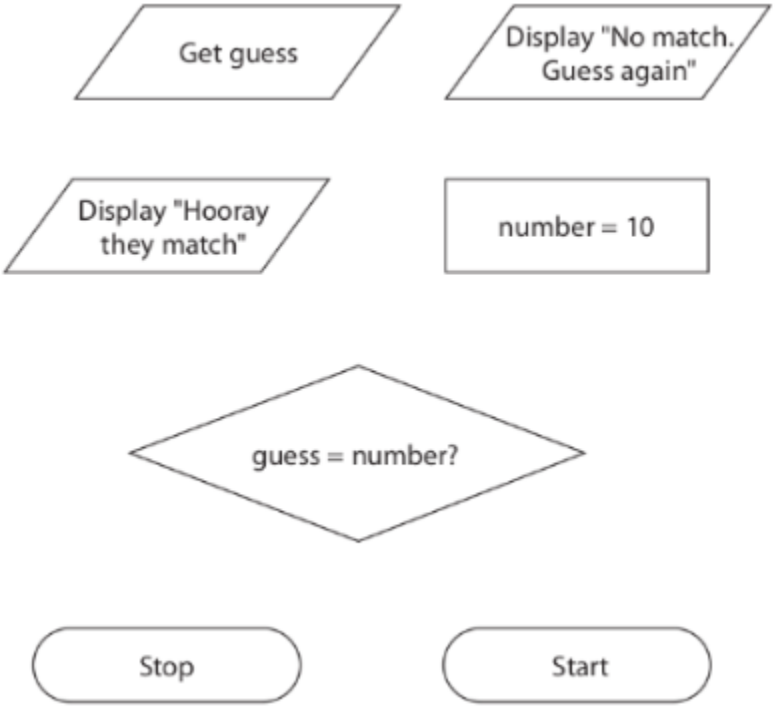
4). (2022 may: paper1:4)

Reba likes writing programs.

(a) She is writing a guessing game.

She needs a flowchart to show the logic of the game.

(i) These are the components needed to draw the flowchart.



Draw the flowchart for the algorithm in the box on the next page.

Use each component once.

Do not add any additional components.

Use as many arrows and yes/no labels as you need. [5]

(ii) Identify an alternative method for writing the algorithm.

- ☐ A Simulation
- ☐ B Cipher
- ☐ C Program code
- ☐ D Truth table

(b) Reba wants to develop a program that will convert a temperature in Fahrenheit to Celsius.

Here are four steps in the algorithm.

The steps are not in the correct order.

	Step
A	Change the temperature to Celsius
B	Get the temperature in Fahrenheit
C	Show the temperature in Celsius
D	Set the temperature to 0

(i) Give the letter of the step that initialises a variable.
(ii) Give the letter of the step that inputs a value.

(c) Figure 2 shows the pseudocode for an early version of an algorithm that Reba has written for another game.
The algorithm:

- asks the user to input a colour or input –1 to end the game
- awards 1 point for red
- awards 8 points for orange
- generates the score for the game
- displays the results of the game.

```
 1 SET Colour TO ""
 2 SET Score TO 0
 3 SET RedPoints TO 0
 4 SET OrangePoints TO 0
 5 SET NumOranges TO 0
 6
 7 WHILE Colour <> "-1" DO
 8    RECEIVE Colour FROM (STRING) KEYBOARD
 9    IF Colour = "red" THEN
10       SET RedPoints TO RedPoints + 1
11    ELSE
12       IF Colour = "orange" THEN
13          SET OrangePoints TO OrangePoints + 8
14          SET NumOranges TO NumOranges + 1
15       END IF
16    END IF
17 END WHILE
18
19 SET Score TO RedPoints + OrangePoints
20
21 SEND ("Score: "& Score) TO DISPLAY
22 SEND ("Number of reds: "& RedPoints) TO DISPLAY
23 SEND ("Number of oranges: "& OrangePoints) TO DISPLAY
```

Figure 2

Reba inputs: red, orange, red, red, orange, –1
The outputs are not as she expects.
(i) Complete the trace table to show the outputs. (4)

Colour	Score	RedPoints	OrangePoints	NumOranges	Outputs
	0	0	0	0	
red					
orange					
red					
red					
orange					
–1					

(ii) Give the line number of the pseudocode that contains the error. (1)
(iii) Write a replacement line of pseudocode to correct the error. (1)

4). (2022 may: paper2:2)
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)

5). (2022 may: paper2:5)
(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)

3	2	10	8	1	9

(ii) Explain one reason why a bubble sort is very efficient in terms of memory usage. (2)
(c) Describe the steps a linear search algorithm takes to find a search item. (3)

6). (2021 Nov:paper 1:4)
4 Algorithms can be used to perform calculations and to process data.
(a) State what is meant by the term algorithm. (1)
(b) Figure 3 shows an algorithm.

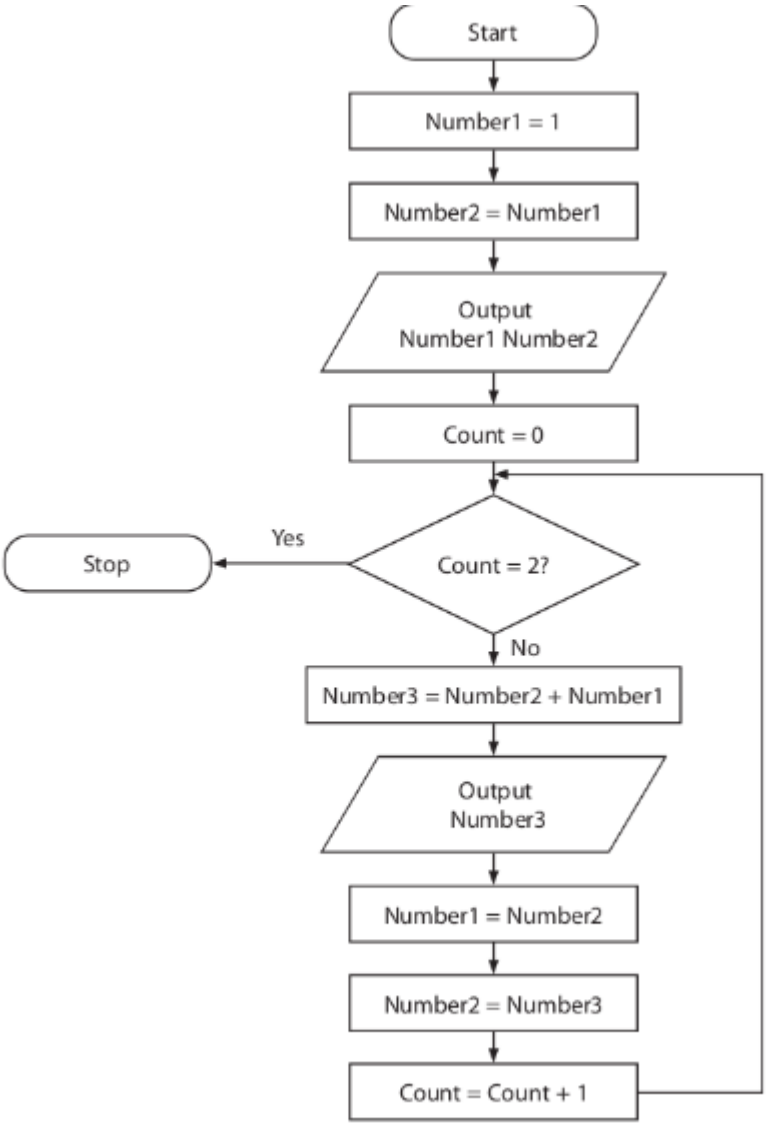


Figure 3

(i) Complete this trace table for the algorithm.

You may not need to use all the rows. (5)

Number1	Number2	Number3	Count	Output	Count = 2?
1	1	–	0	1 1	False

(ii) The benefits of using a trace table include that they allow variable states, outputs and decisions to be recorded.
Give one other benefit of using a trace table to test an algorithm. (1)

(c) An algorithm is needed to count and display the number of vowels in a word.

The vowels are a, e, i, o, and u.

The completed algorithm must:

•count the number of vowels

•create a message as a single string (e.g. there are number vowels)

•print the message to the display.

Here is a partially completed algorithm written in pseudocode.

Complete the algorithm in the spaces provided. (5)

```
SET word TO 'elephant'

FOR EACH letter FROM word DO

END FOREACH
```

7). (2021 Nov:paper 2:1)

(c) **Figure 1** shows a programming construct in a flowchart. (1)

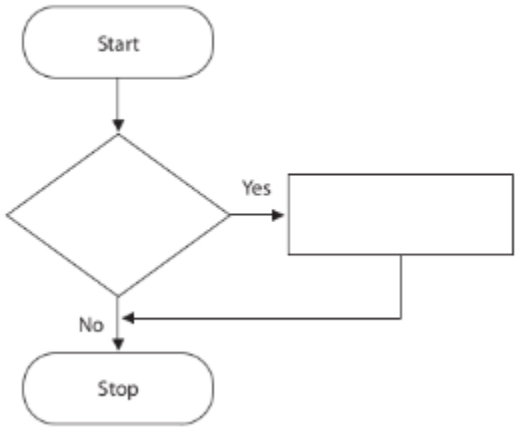


Figure 1
Identify the name of the programming construct.
☐ A Array
☐ B Iteration
☐ C Operator
☐ D Selection

8). (2021 Nov:paper 1:5)
Isaac is a program developer.
(a) Figure 3 shows an algorithm Isaac has written.

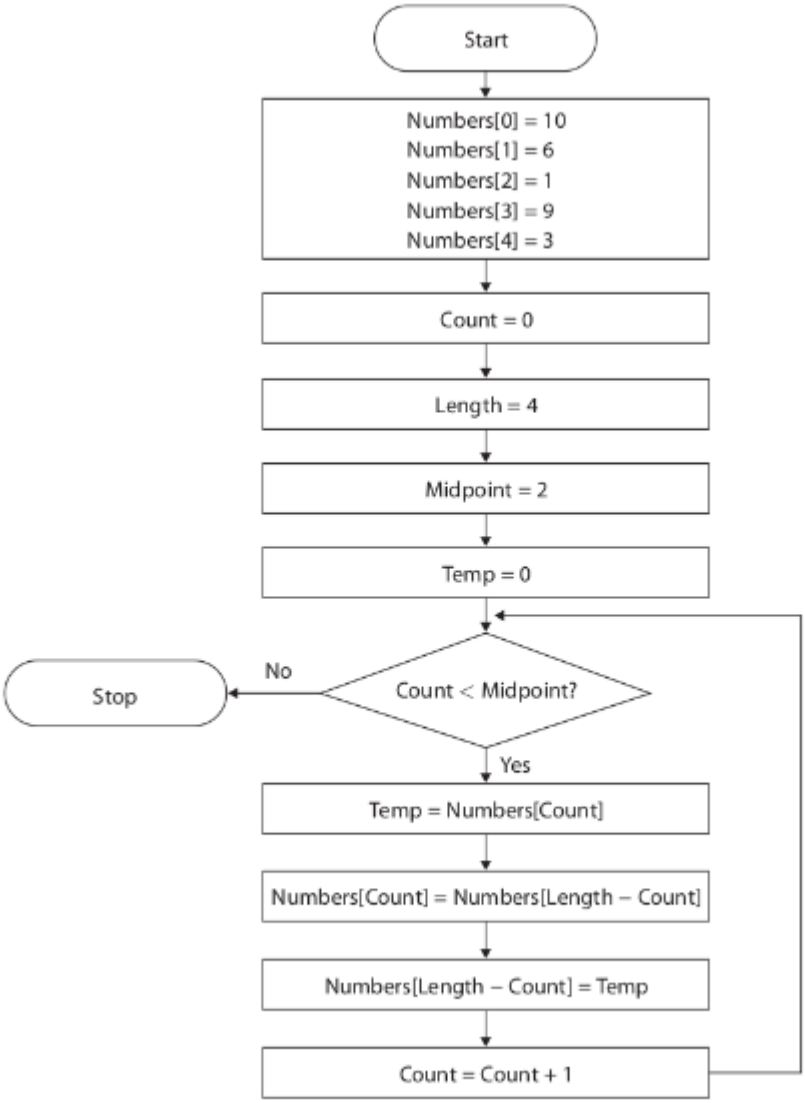


Figure 3

(i) Complete the trace table. You may not need to use all of the rows. (5)
(ii) Give the purpose of the algorithm. (1)
(iii) Explain why the variable Temp is needed. (2)

				Numbers array				
Count	Length	Midpoint	Temp	[0]	[1]	[2]	[3]	[4]
0	4	2	0	10	6	1	9	3

(b) Figure 4 shows an algorithm Isaac has written using pseudocode.
The algorithm should display the average of the numbers that have been input.

```
1   SET total TO 0
2   SET number TO 0
3   SET count TO 0
4   WHILE number <> –1 DO
5       SEND 'Input a number or –1 to end the program' TO DISPLAY
6       RECEIVE number FROM (INTEGER) KEYBOARD
7       SET total TO total + number
8       SET count TO count + 1
9   END WHILE
10  SET average TO total / count
11  SEND 'The average is ' & average TO DISPLAY
```

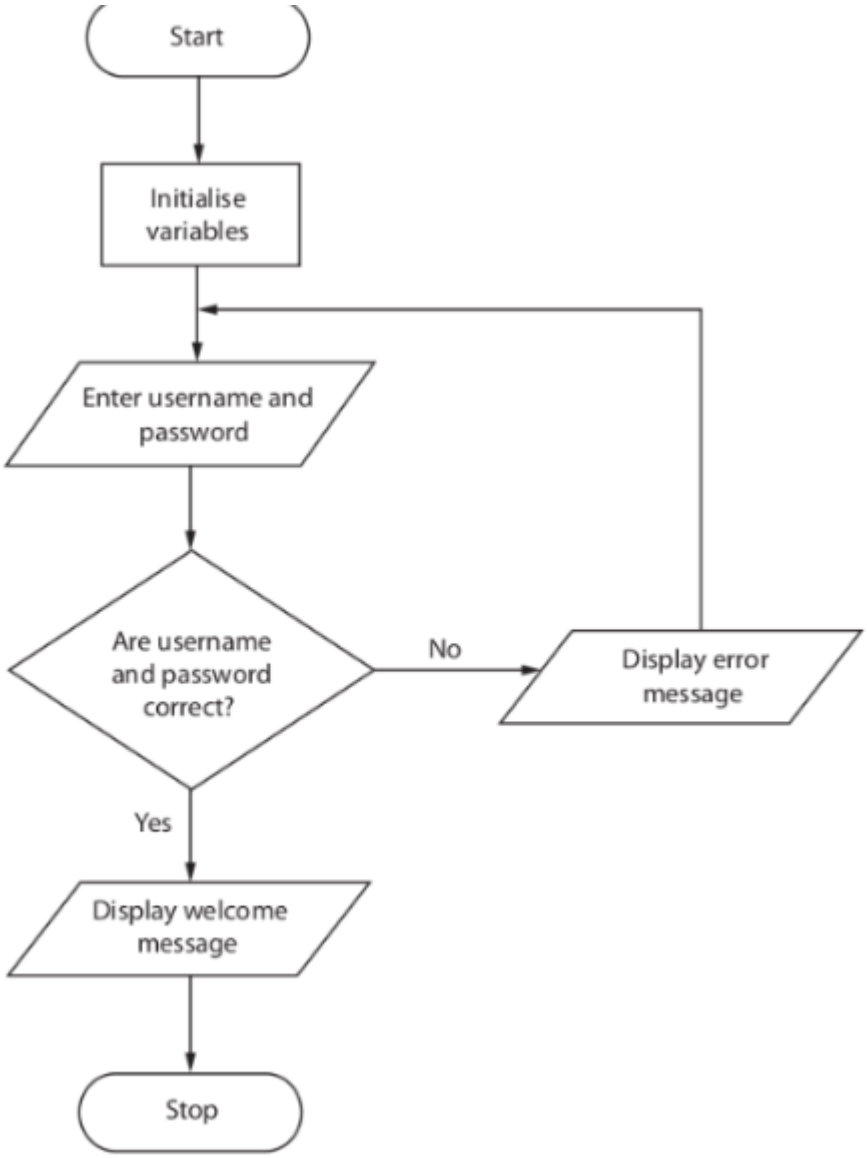
Figure 4
Isaac uses the input 2, 3, 5, 2, -1 to test the algorithm. He discovers an error

Expected result	Actual result
The average is 2.75	The average is 2.2

(i) Explain why the Actual result is not the same as the Expected result. (2)
(ii) Give the number of the line that contains the error. (1)
(iii) Amend a single line of pseudocode to correct the error. (1)
9). (2021 Nov:paper 2:2)
A book club uses computer applications.
(a) The club wants a program to ensure that logins are valid.
A flowchart for a test version of the program has been designed.
The test version uses:

- a username of bard423
- a password of nX2934?

Here is the flowchart.



Explain the Program [5]

10). (2021 Nov:paper 2:4)

Zak sells snacks at an after-school club.

He wants a program that will hold details of the products he sells.

The program must:

•

•

generate a five-character product code that:

–starts with the first three letters of the product name

–ends with a random number between 10 and 30

display the product code followed by the product name.

(b) Zak plans to implement a binary search algorithm to search a table of products.

(i) Explain one advantage of a binary search compared to a linear search. (2)

(ii) Figure 2 shows an algorithm for a binary search.

```
1     #     Initialise variables
2     SET productSearch TO " "
3     SET startPosition TO 1
4     SET midPosition TO 0
5     SET endPosition TO LENGTH(productList)
6     SET found TO False
7
8     #     Print prompts, take and check input from user
9     SEND "Enter the product code" TO DISPLAY
10    RECEIVE productSearch FROM (STRING) KEYBOARD
11    REPEAT
12        midPosition = (startPosition + endPosition) DIV 2
13        IF productList[midPosition] < productSearch THEN
14            startPosition = midPosition + 1
15        END IF
16        IF productList[midPosition] > productSearch THEN
17            endPosition = midPosition - 1
18        END IF
19    UNTIL productList[midPosition] = productSearch OR startPosition = endPosition
```

Figure 2

This binary search algorithm will be used to search the product list for the product code str15.

Complete the table to indicate the order in which the product codes will be examined by the algorithm.

Write the number 1 by the first product code to be examined, 2 by the second code to be examined, and so on.

Position in list	Product code	Order examined
1	ark11	
2	asp11	
3	bar13	
4	dri15	
5	mil19	
6	rib10	
7	str15	
8	tor16	

(iii) Zak has another list containing the names of five students who attend the

after-school club.

Give the maximum number of names that would need to be examined by the binary search algorithm to determine whether a name appears in the list. (1)

(iv) Give the name of an algorithm that could be used to sort a list. (1)