Computing @ Charlton

GCSE Computer Science

Mark Scheme

Exam Practice

Topic 1: Computational thinking

This booklet contains *all* past questions on this topic

Topic 1: Computational thinking

Students are expected to develop a set of computational thinking skills that enable them to design, implement and analyse algorithms for solving problems.

Students are expected to be familiar with and use the Programming Language Subset (PLS) document provided on the GCSE in Computer Science section of our website.

The flowchart symbols students are expected to be familiar with and use are shown in *Appendix 2.*

Subject content	Students should:	
1.1 Decomposition and abstraction	1.1.1	understand the benefit of using decomposition and abstraction to model aspects of the real world and analyse, understand and solve problems
	1.1.2	understand the benefits of using subprograms
1.2 Algorithms	1.2.1	be able to follow and write algorithms (flowcharts, pseudocode*, program code) that use sequence, selection, repetition (count-controlled, condition-controlled) and iteration (over every item in a data structure), and input, processing and output to solve problems
	1.2.2	understand the need for and be able to follow and write algorithms that use variables and constants and one- and two-dimensional data structures (strings, records, arrays)
	1.2.3	understand the need for and be able to follow and write algorithms that use arithmetic operators (addition, subtraction, division, multiplication, modulus, integer division, exponentiation), relational operators (equal to, less than, greater than, not equal to, less than or equal to, greater than or equal to) and logical operators (AND, OR, NOT)
	1.2.4	be able to determine the correct output of an algorithm for a given set of data and use a trace table to determine what value a variable will hold at a given point in an algorithm
	1.2.5	understand types of errors that can occur in programs (syntax, logic, runtime) and be able to identify and correct logic errors in algorithms
	1.2.6	understand how standard algorithms (bubble sort, merge sort, linear search, binary search) work
	1.2.7	be able to use logical reasoning and test data to evaluate an algorithm's fitness for purpose and efficiency (number of compares, number of passes through a loop, use of memory)
1.3 Truth tables	1.3.1	be able to apply logical operators (AND, OR, NOT) in truth tables with up to three inputs to solve problems

*In this specification, the term 'pseudocode' is used to denote an informal written description of a program. Pseudocode uses imprecise English language statements and does not require any strict programming syntax.

Answer ALL questions. Write your answers in the spaces provided.

Some questions must be answered with a cross in a box ☒. If you change your mind about an answer, put a line through the box ☒ and then mark your new answer with a cross ☒.

1 Computational thinking

(a) Identify the term that means breaking a problem or solution down into smaller parts.

(1)

☒ **A** Abstraction

☒ **B** Computation

☒ **C** Decomposition

☒ **D** Evaluation

The only correct answer is C *A is not correct because abstraction is hiding or removing detail* *B is not correct because computation is the act of computing* *D is not correct because evaluation is reaching a conclusion*

(b) State **two benefits** of subprograms.

(2)

1 ....

2 ....

Award **one** mark for any of the following up to a maximum of **two** marks: • The subprogram code only has to be written once / saves development time (1) • The subprogram can be called many times (reusability) (1) • The subprogram only has to be debugged once (1) • It is easier to locate and debug errors (1) • Hides the details of how a function/procedure/code works (abstraction) (1) • Can be used by programmers who don't have the knowledge to write them (1) • Can be grouped into libraries/can be shared with other programmers (1) • Makes programs easier to read/understand (1) • Reduces the overall size of code (1) • Problem can be decomposed into smaller sub-problems (1) • Allows a team to work on a project (1)	Do not accept generalisations such as easier/faster/quicker without qualification

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

(c) Here is an algorithm that uses colours.

```
 1  # ------------------------------------------------------------
 2  # Global variables
 3  # ------------------------------------------------------------
 4  theColours = ["Green", "Blue", "Yellow", "Red", "Purple"]
 5  colour = ""
 6
 7  # ------------------------------------------------------------
 8  # Main program
 9  # ------------------------------------------------------------
10
11  for item in theColours:
12      print (item)
13
14  colour = input ("Enter a colour: ")
15  while (colour != ""):
16      if (colour == "Green"):
17          print ("Green is my favourite colour")
18      else:
19          print (colour + " is a good colour")
20
21      colour = input ("Enter a colour: ")
```

(i) Give the first line number of a condition-controlled loop.

(1)

- 15 (1)

(ii) Give the first line number of iteration over every item in a data structure.

(1)

- 11 (1)

(iii) Give the line numbers of a selection.

(1)

Award **one** mark for any of the following: • 16, 17, 18, 19 (1) • 16 - 19 (1) • 16, 18 (1)	Do not accept 16 on its own.

(d) Programs can have syntax errors and runtime errors.

(i) Define the term 'syntax error'.

	Guidance
Award **one** mark for any of the following: • Code that breaks/violates the rules/grammar of the programming **language** (1)	Do not award credit for an example in isolation such as missing colon, there must be a definition.

~~(ii) Runtime errors happen when a program is executing.~~

Explain a runtime error.

	Guidance
Award up to **two** marks for a linked explanation, such as: • The program crashes/stops (1) because the operation the computer is asked to do is impossible (1) • The program crashes (1) because the CPU cannot execute one of the instructions in the code (1)	For both marks, the expansion must follow/associate with the statement. Do not award credit for an example in isolation such as division by zero. Be careful not to award marks for syntax/logic error

(e) Algorithms use relational and arithmetic operators.

(i) Here is a relational operator used in a conditional test.

```
count > index
```

State the **two** different results of evaluating a conditional test.

1

2

	Guidance
Award **one** mark for each: • True (1) • False (1)	Do not award 0, 1, Yes, No Allow T/F for True/False

(ii) Identify the result of 5 // 2

(1)

☒ **A** 0.5

☒ **B** 1

☒ **C** 2

☒ **D** 2.5

The only correct answer is C *A is not correct because 0.5 is the fractional part of division* *B is not correct because 1 is the remainder after integer division, i.e. the result of modulus* *D is not correct because 2.5 is the result of division*

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

(f) Programmers consider algorithm efficiency when they write code.

(i) Sorting and searching use algorithms.

Complete the table with the name of a search algorithm and a sort algorithm.

(2)

Algorithm type	**Characteristic**	**Algorithm name**
Search	Is a divide and conquer algorithm	Binary search (1)
Sort	Is **not** a divide and conquer algorithm	Bubble sort (1)

Ignore exclusion of the words 'search' and 'sort' in the last column

(ii) Explain **one** effect the number of comparisons has on the execution time of a sorting algorithm.

(2)

Award up to **two** marks for a linked explanation, such as: • A sorting algorithm executes more quickly (1) because a small number of comparisons are made (1) • A sorting algorithm executes more slowly (1) because a large number of comparisons are made (1)	For both marks, the expansion must follow/associate with the statement.

(Total for Question 1 = 16 marks)

4 Computational thinking

(a) Programmers use trace tables with algorithms.

Explain the purpose of a trace table.

(2)

Award up to **two** marks for a linked explanation, such as: • It helps the programmer visualise the steps in a program / find errors/bugs / check the algorithm gives the correct output (1) because they can see the value a variable holds at a given point in an algorithm (1)	For both marks, the expansion must associate with the statement.

(b) Algorithms use arrays and records to hold data.

Describe a record.

(2)

Award up to **two** marks for a linked description, such as: • A list / data structure / collection of elements/fields/values (1) where each element/field/value may be a different data type / is related to the others (1)

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

BLANK PAGE

(c) An algorithm allows users to enter a whole number.

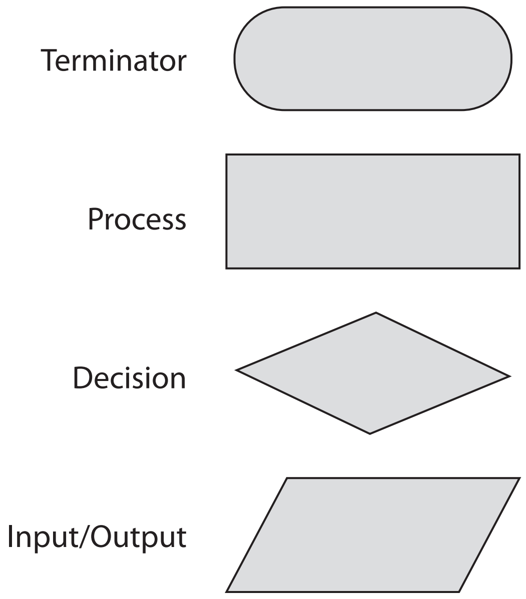
The number can be positive or negative.

The purpose of the algorithm is to report whether the number is even or odd.

The modulus function returns the remainder after division.

The algorithm can be expressed as a flowchart.

Here are some flowchart symbols:



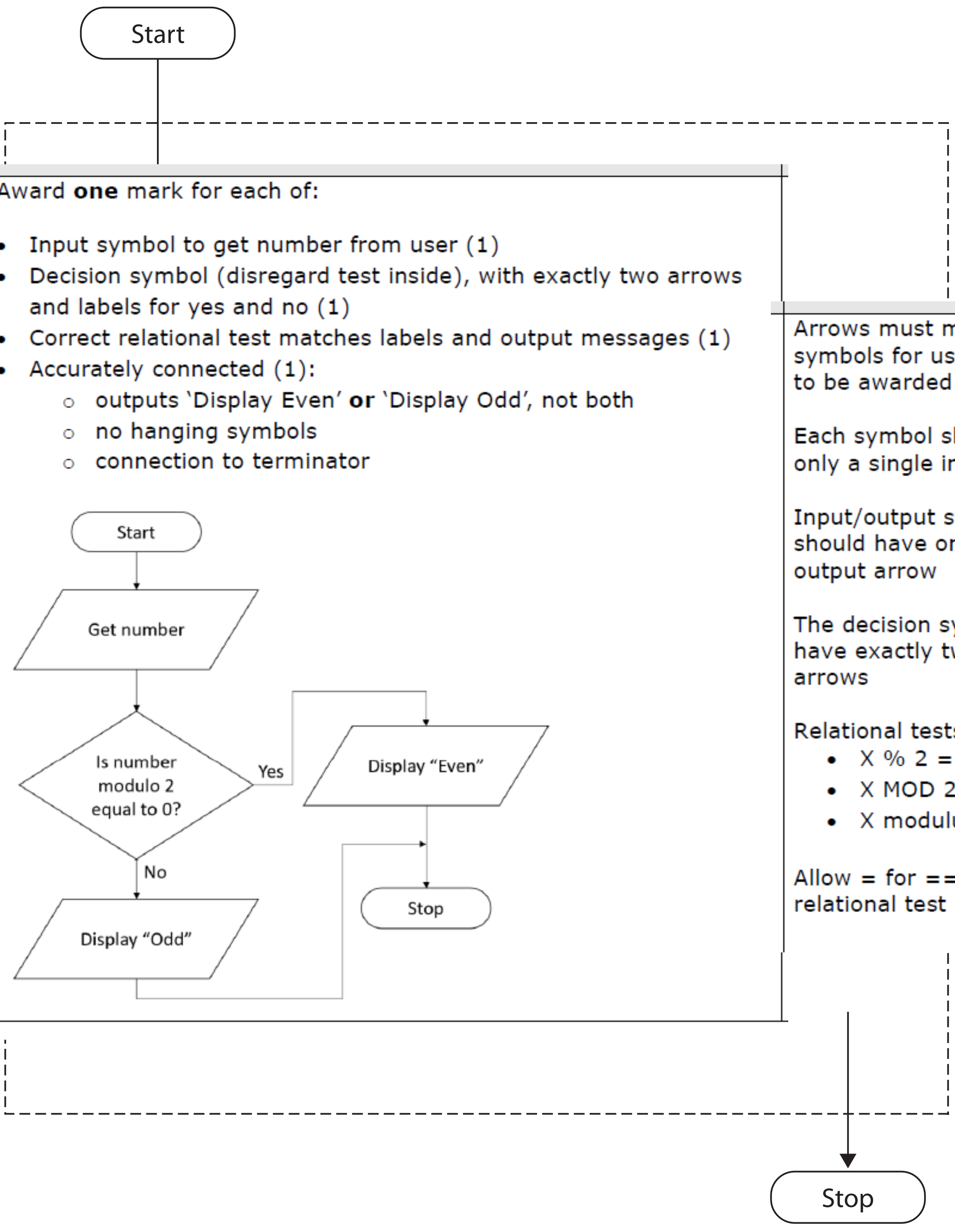
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

Complete the flowchart to show the algorithm.

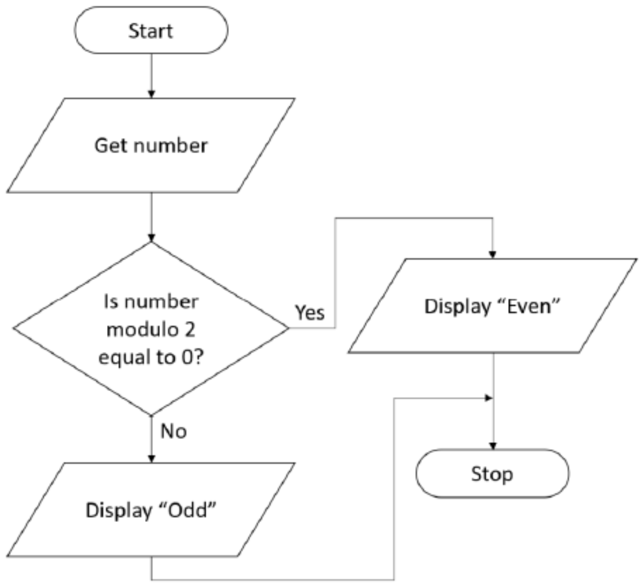
You may not need to use all the flowchart symbols.

(4)



Award **one** mark for each of:

- Input symbol to get number from user (1)
- Decision symbol (disregard test inside), with exactly two arrows and labels for yes and no (1)
- Correct relational test matches labels and output messages (1)
- Accurately connected (1):
    - outputs 'Display Even' **or** 'Display Odd', not both
    - no hanging symbols
    - connection to terminator



Arrows must match symbols for use of symbol to be awarded

Each symbol should have only a single input arrow

Input/output symbols should have only a single output arrow

The decision symbol must have exactly two output arrows

Relational tests:

- X % 2 == 0
- X MOD 2 == 0
- X modulus 2 == 0

Allow = for == in relational test

(d) A linear search algorithm can be used on both a sorted and an unsorted array.

Describe how a linear search algorithm operates on an **unsorted** array.

(4)

Award up to **four** marks for a linked description, such as:

- Start at the first position / Iterate/Traverse (through the array) (1), compare the item with the target (1), stop when the target is matched (1), or stop when the end of the list is reached (and the item is not matched) (1)

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

(e) Algorithms control physical devices using logical operators.

A security system turns on a floodlight when the sunlight falls below a certain level (S) and a movement sensor is activated (M).

Complete the truth table.

(2)

Example:

S	M	S AND M
0	0	0
0	1	0
1	0	0
1	1	1

Award **one** mark for each of:

- Four unique rows for the values of S and M, indicating binary representation of 0, 1, 2, and 3 (1)
- Four correct results of applying AND to the values in S and M (1)

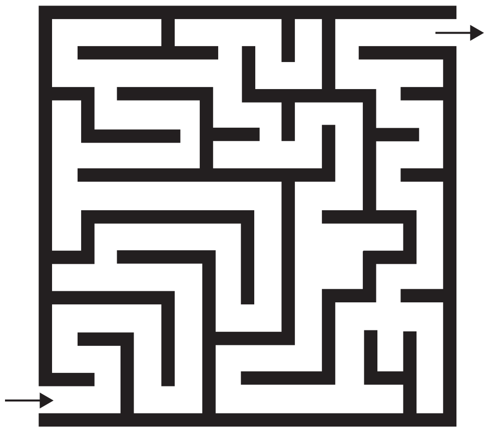
Ignore order of rows

Award both marks only if completely correct

Award carry through in final column, if error in S or M columns

Only award:
- 1/T/True/On
- 0/F/False/Off

(f) A group of students are working together on a single maze game. The arrow keys control the character. When the character reaches the end of the maze without touching a wall, a happy sound is played. The game also displays a score.



Discuss the use of decomposition and abstraction in developing this game.

Your answer should include:

- a definition of each term
- the benefits each brings to the group of students
- an example of where each could appear in the program code.

(6)

Indicative content:

Definition

- Decomposition is breaking down into smaller parts. Problems, solutions, and algorithms can be decomposed.
- Abstraction is the process of removing or hiding unnecessary detail.

Benefits

- It is usually easier to solve several smaller problems, such as checking if touching a wall or updating the score display, than solve one big problem, such as making a game.
- Different parts of the program can be shared between the class members to speed up development, for example, one group could work on the code to control the character, while another works on creating and playing the sounds.
- Once all the pieces, like sounds, movement, and score are working correctly, the smaller solutions can be combined to make a larger solution, with fewer errors.
- The individual parts of the program, such as updating the score can be ignored by the group of students writing the code for moving the character with the arrows / allowing each group of students to focus only on the small problem they have been given means time is not wasted on analysis not relevant to the solution.

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

Appear in program code

- Different blocks of code logic show decomposition, such as moving the character and updating the score. These blocks could be shown separated by white space.
- Subprograms are decompositions because they're blocks of code logic, separated from the main program. Subprograms could be used for updating the score and resetting the character back to the starting position.
- Abstraction is used each time a subprogram, either built-in, in a library, or in the code file is called. Library routines in the game would include one to play the sounds and to get the keyboard input.
- Subprograms are abstractions because their names hide how they work internally, even if their name is not descriptive. A subprogram to update the score should have a descriptive name, such as updateScore, but could just be called X. Either way, how it works internally is still hidden from the caller.

Level	Mark	Descriptor
	0	No rewardable content.
Level 1	1–2	Basic, independent points are made, showing elements of understanding of key concepts/principles of computer science. (AO1) The discussion will contain basic information with little linkage between points made or application to the context. (AO2)
Level 2	3–4	Demonstrates adequate understanding of key concepts/principles of computer science. (AO1) The discussion shows some linkages and lines of reasoning with some structure and application to the context. (AO2)
Level 3	5–6	Demonstrates comprehensive understanding of key concepts/principles of computer science to support the discussion being presented. (AO1) The discussion is well developed, with sustained lines of reasoning that are coherent and logically structured, and which clearly apply to the context. (AO2)

(Total for Question 4 = 20 marks)

5 Computational thinking

(a) Here is an algorithm that prints colours.

```
# ---------- Global variables ----------
inkColours = ["Cyan", "Magenta", "Yellow", "Black"]

# ---------- Subprograms ----------
def displayAll (inList):
    for index in range (0, len (inList)):
        print (index, inList[index])

# ---------- Main Program ----------
userInput = input ("Would you like to see the colours? ")
if ((userInput == "y") or (userInput == "Y")):
    displayAll (inkColours)
else:
    print ("Thank you")
```

(i) State the type of data structure used to hold the ink colours. **(1)**

Number	
5(a)(i) **Clerical**	Any **one** from: • Array • List

(ii) Give the contents of `inkColours[2]` **(1)**

Yellow

(iii) State the name of the computational thinking technique used by the subprogram `displayAll()` to hide the logic of printing. **(1)**

Abstraction

(iv) State the name of the computational thinking technique used when separating logic into different blocks, such as the subprogram and the main program. **(1)**

Decomposition

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA

(b) This algorithm searches a sorted array of numbers for a target value.
The target value may or may not be in the array.

```
while ((index < len (theArray)) and (not found) and (not passed)):
    if (theArray[index] == theTarget):
        found = True
        location = index
    elif (theArray[index] > theTarget):
        passed = True
    index = index + 1
```

The use of the `found` variable helps to make the algorithm efficient.

Describe how the use of the `passed` variable also helps to make the algorithm efficient.

(2)

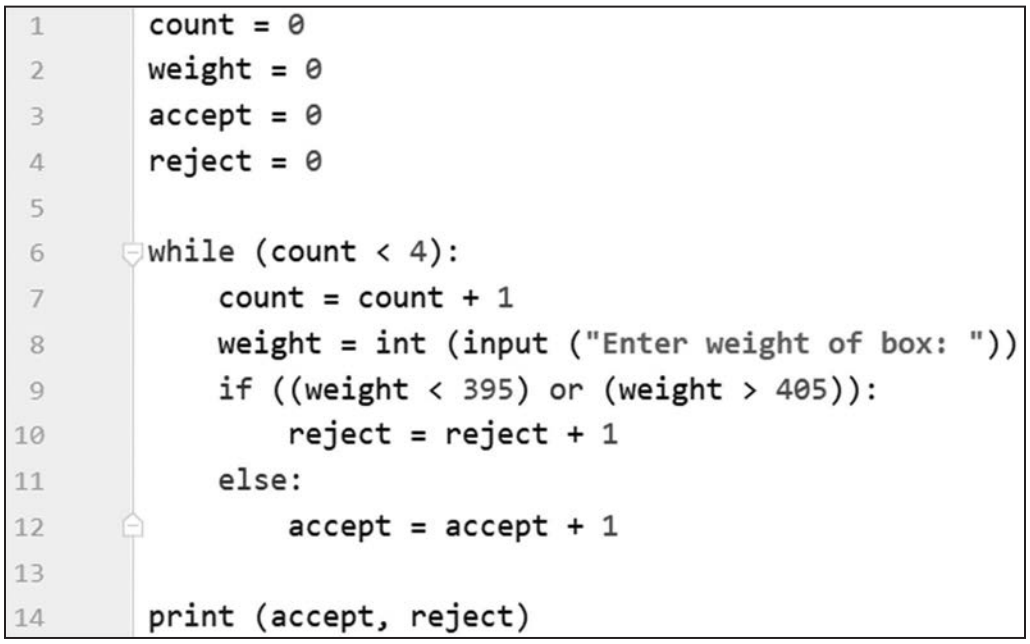
A linked description such as:

- If the value of the item at the current index position in the array is more than the value of the target (1), no more passes of the loop are required / the loop will exit (1)
- If the pass through the loop goes past the expected location of the target value (1), the third condition for the loop to execute will not be met (1)
- If the value (of 'theTarget') is exceeded without reaching the end of the array (1) the algorithm will end (1)

Turn over ▶

(c) Laura owns a fruit shop.

This program checks the weight of boxes of strawberries from Laura's shop.



```
count = 0
weight = 0
accept = 0
reject = 0

while (count < 4):
    count = count + 1
    weight = int (input ("Enter weight of box: "))
    if ((weight < 395) or (weight > 405)):
        reject = reject + 1
    else:
        accept = accept + 1

print (accept, reject)
```

DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

The inputs are 404, 393, 395, 405.

Complete the trace table showing the execution of the program with these inputs.

You may not need to fill in all the rows in the table.

(6)

count	accept	reject	weight	Display

One mark for each correct row showing relationship of variable changes.

count	accept	reject	weight	Display
0	0	0	0	
1	1		404	
2		1	393	
3	2		395	
4	3		405	
				3 1

Different versions of trace tables are acceptable.

Blanks can be replaced with contents of previous row.

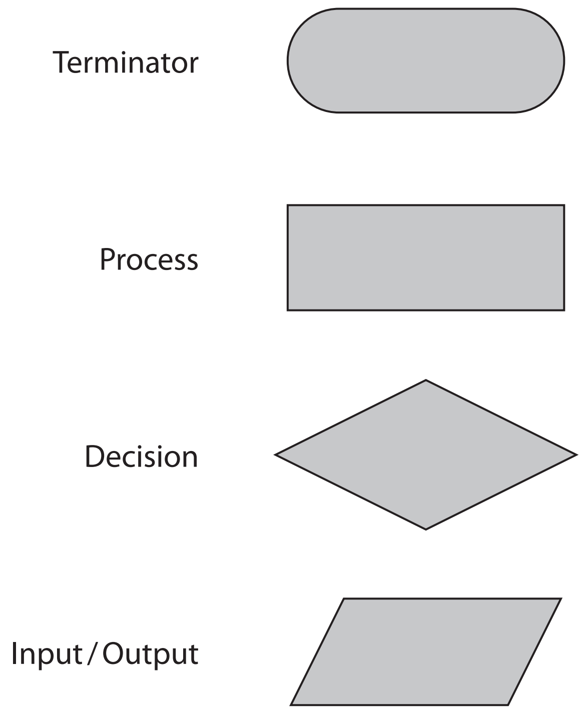
Final '3 1' can be included on row above.

Ignore formatting of 3 1 in Display column (e.g. accept '3,1')

Allow 'Enter weight of box:' in Display column until 3 1 should be displayed.

(d) An algorithm is required that allows a user to enter two numbers. The algorithm then informs the user which number is greater, or whether the two numbers are equal. The algorithm is expressed in a flowchart.

Here are some flowchart symbols:

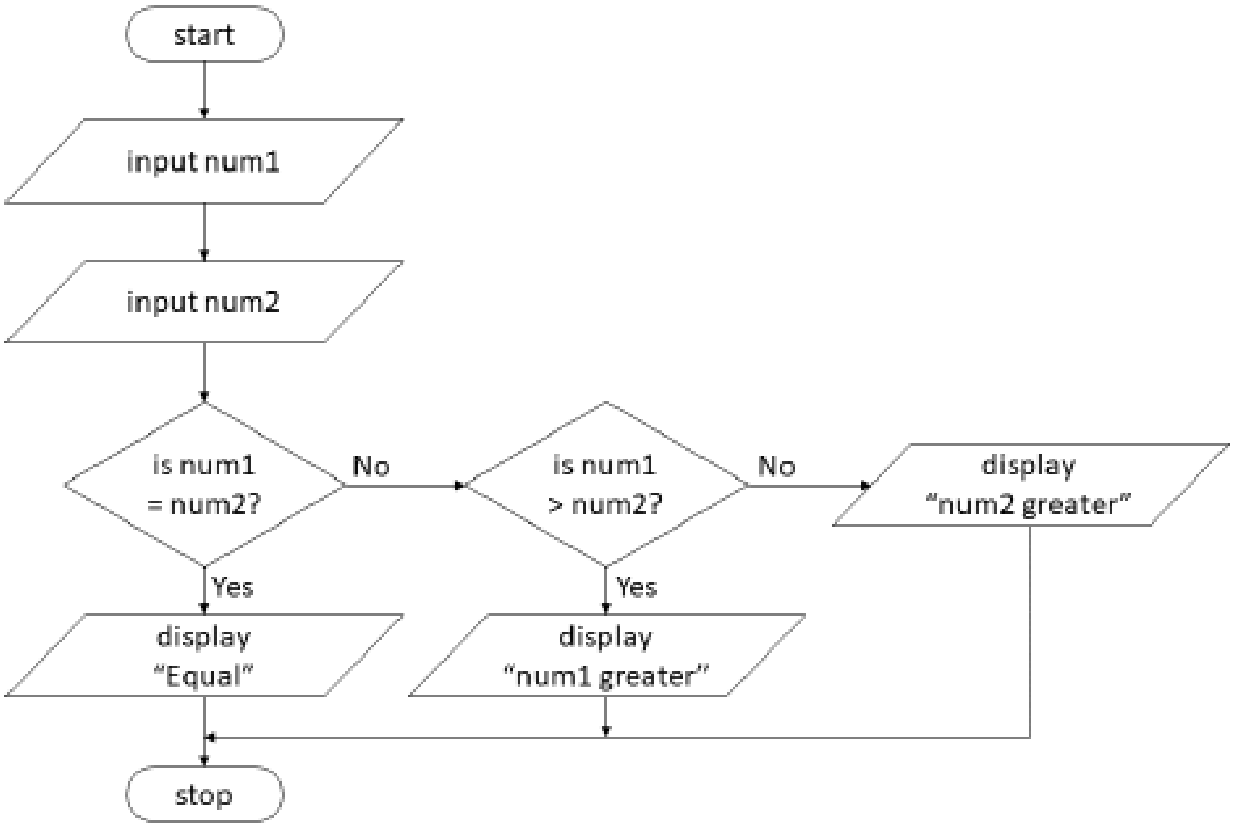


DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

Draw a flowchart to show this algorithm.

(6)



Additional Guidance
• Award 'End', 'Stop', 'Start' and 'Begin' as text for terminator symbols. • Award '==' and '=' used for equivalence inside decision symbol. • Accept 'Print' or 'Output' as an alternative to 'Display' in the output symbols. No quotes required around output string. • Accept True/False for Yes/No labels

(Total for Question 5 = 18 marks)

TOTAL FOR PAPER = 75 MARKS

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

3 Computational thinking

(a) Define the term 'decomposition'. **(1)**

Breaking down a problem/solution/system/algorithm

(b) State the worst case for a linear search algorithm. **(1)**

The target item is not in the list / is at end of the list

(c) A bubble sort is carried out on this list to put it in ascending order.

8 3 2 4 0 3 9

The value '8' starts in position 0.

(i) State the number of passes required to complete the sort. **(1)**

5

(ii) State the number of swaps made on the final pass. **(1)**

0

(iii) State the component of an algorithm used to store whether a swap has been made during a pass.

Variable	Accept words that describe a use of variable in this case, e.g. flag

(iv) State the position of the item that will be compared with the value in position 0. **(1)**

1

(v) Define the term 'iteration'. **(1)**

Looping over every item in a data structure

(d) Explain **one** benefit of using subprograms.

(2)

An explanation to include **two** from:

- The subprogram may be used more than once in a program (1) so that writing, debugging, testing will save time (1)
- The subprogram performs one specific/contained task (1) so it can be moved away from the main program code (1)
- Subprograms for common tasks can be stored in libraries and reused in other programs (1) so that they don't have to be re-written (1)

(e) Complete this truth table.

(3)

A	B	C	(A AND B)	(NOT C)	(A AND B) OR (NOT C)
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	0	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	1	0	1

Additional Guidance
Allow follow through

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

(f) The identifier `plants` is used for an array of values.

`len(plants) // 2` is used to find the index position of the middle item in `plants`

Explain **one** reason why integer division, rather than division, is used to do this.

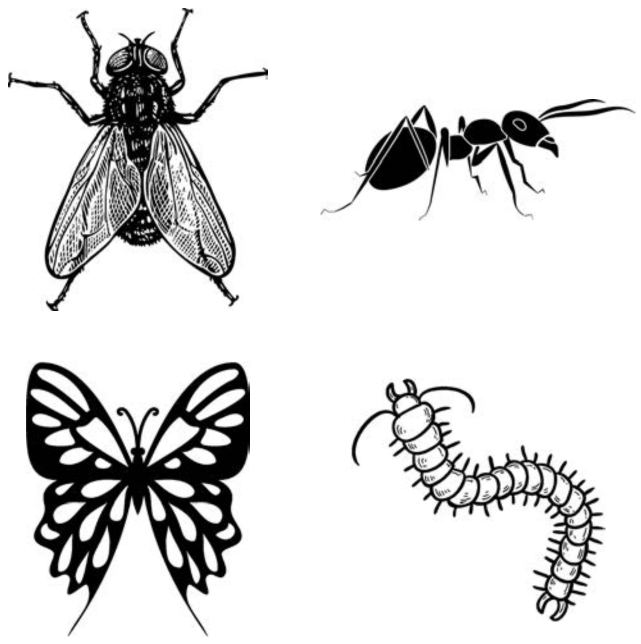
(2)

An explanation to include **two** from:
An integer is returned with `//` (1) because

- `len(plants)` could be an odd number (1)
- the value for position has to be an integer / not a decimal (1)

e.g.
If the length of the array is an odd number, normal division will return a real number (1) (which is not acceptable because) index values are integers (1)

(g) Here are four images of creatures that will be used in a computer game.



(Source: images from © PAL)

Before coding the game, a programmer applies abstraction.

One feature of creatures is their colour.

State **two other** features of the creatures that the programmer could include when creating a general model for a creature.

(2)

1 ......

2 ......

Any **two** from: • Number of legs • Number of antennae • Length / width / size • Number of wings • Size of head • Shape of body • Number of eyes • Size of mouth • Habitat

Additional Guidance
Accept any appropriate feature. Award two marks from the same bullet point for the identification of two different features e.g. number of legs (1), legs jointed/unjointed (1)

(Total for Question 3 = 16 marks)

DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA
DO NOT WRITE IN THIS AREA

2 Computational thinking

(a) Programmers use abstraction to model the real world.

Define the term 'abstraction'.

(2)

A definition to include:

- The process of removing or hiding unnecessary (1) details so that only the important points remain (1)
- Programmers can focus only on the important details of a problem (1) because abstraction allows them to ignore (1) any detail that is not relevant.

(b) Programmers use different types of operators in their programs.

Name the **type** of operator for each example.

(3)

<, !=

+, *

AND, NOT

- Relational (1)
- Arithmetic (1)
- Boolean/Logical (1)

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

(c) Here is an algorithm.

```
myNumber = 0

myNumber = int (input ("Enter a whole number between 1 and 100:"))

if (myNumber < 1):
    print ("Too low")
elif (myNumber > 100):
    print ("Too high")
elif (myNumber % 10 == 0):
    print ("Nice round number")
elif (myNumber == 100):
    print ("That's the biggest number")
else:
    print ("Good choice")
```

Complete the table to show the output for the given input.

(4)

Input	Output
200	Too high (1)
33	Good choice (1)
100	Nice round number (1)
0	Too low (1)

(d) Describe **one** difference between a syntax error and a logic error.

(2)

A description to include:

- A syntax error is caused by using the grammar/form/words of the programming language incorrectly (1), whereas a logic error is caused by an error in the design of the algorithm (1)
- A syntax error causes the program not to interpret/compile/translate, whereas a logic error produces an incorrect/wrong result (1)

(e) A merge sort is carried out on this list.

9 4 3 5 5 1 7

(i) State the number of splits required to complete the sort.

Six/6

(1)

(ii) Here is the list after being split.

Complete the diagram to show the merging steps.

(2)

9	4	3	5	5	1	7

The list has already been split into 7 individual lists. The response here should only show the missing merges.

One mark for each correct merge:

4 9	3 5	1 5	7	(1)
3 4 5 9	1 5 7			(1)

1	3	4	5	5	7	9

(iii) Explain the effect on efficiency of using a merge sort algorithm instead of a bubble sort algorithm on the original list.

9 4 3 5 5 1 7

(2)

An explanation to include:

- The merge sort will use more memory (1) because it makes copies of the list (1).
- The bubble sort will use less memory (1) because it is an in-place (1) sort.
- The merge sort will take more time to write and debug (1), because it is a much more complex (1) algorithm.
- The bubble sort will probably take less time to write (1) because it is a simple/well-known (1) algorithm.

16 marks)

Answer ALL questions. Write your answers in the spaces provided.

Some questions must be answered with a cross in a box ☒. If you change your mind about an answer, put a line through the box ☒ and then mark your new answer with a cross ☒.

1 Computational thinking

(a) An algorithm is designed for a 'guess the roll of a dice' game. The player decides the number of sides on a single dice, between 6 and 12, and guesses the number rolled.

© PAL

(i) Name **two** inputs required by the algorithm.

(2)

1. • The number of sides on the single dice/a number between 6 and 12 (1) • The user's guess of the roll (1) 2.	• Award variable names provided they are meaningful and uniquely identify the inputs, e.g. theGuess, sides.

(ii) The player of the game can have three guesses. The variable `roll` is set to 1 at the start of the game. One is added to `roll` after each guess.

Identify the statement with the correct relational operator used to check whether the player can have another guess.

(1)

☒ **A** `roll = 3`

☒ **B** `roll == 3`

☒ **C** `roll > 3`

☒ **D** `roll <= 3`

D roll <= 3

- A roll = 3 is not a relational operator, but an assignment
- B roll == 3 is an equivalence check, which would not allow a guess for 1 and 2
- C roll > 3 is greater than, which would not allow a guess for 1, 2, or 3

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

(b) Complete the truth table for the logical expression NOT (A OR B).

(2)

One mark for each correct column

A	B	A OR B	NOT (A OR B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

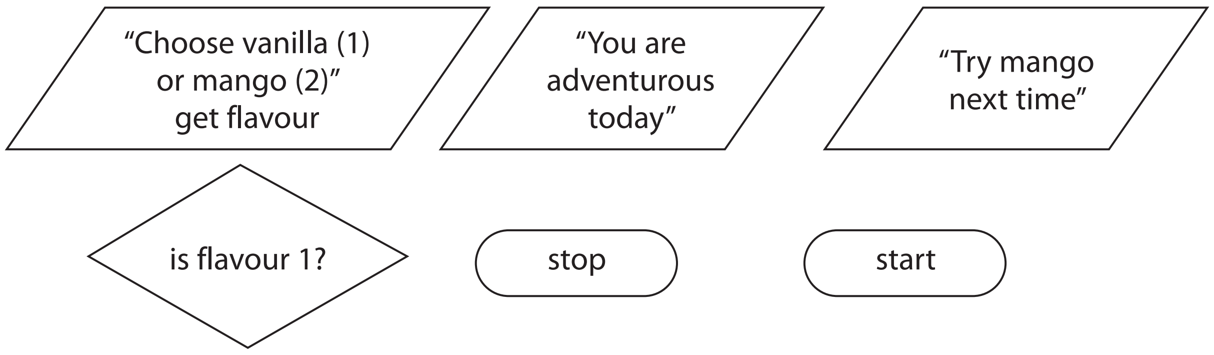
(c) Define the term 'subprogram'.

(2)

One mark each for a maximum of 2

- A subprogram is a self-contained block (1) of code.
- A subprogram performs a specific/dedicated task (1).
- It can be 'called' by the main program or other subprograms, when it is needed (1).

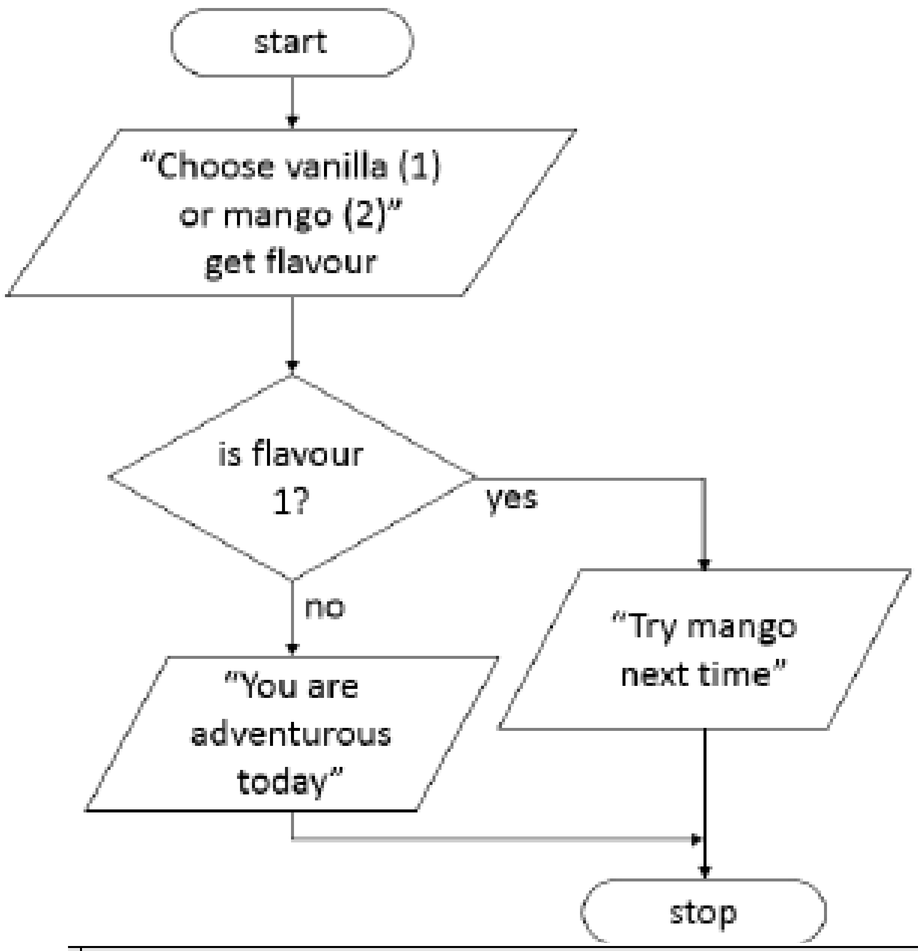
(d) Here is a set of flowchart symbols for an algorithm about flavours.


"Choose vanilla (1) or mango (2)" get flavour
"You are adventurous today"
"Try mango next time"
is flavour 1?
stop
start

Draw a flowchart to show the completed algorithm.

Use every symbol exactly once. Use as many arrows and labels as you require.
Do **not** add any additional symbols.

(4)


start
"Choose vanilla (1) or mango (2)" get flavour
is flavour 1?
yes
no
"Try mango next time"
"You are adventurous today"
stop

Answer

- Start and stop terminators in correct positions (1)
- Decision symbol has two outputs only, i.e. the two output messages (1)
- Yes/no labels on decision match output messages (1)
- Fully connected to function correctly (1)

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

(e) State **three** types of error that can occur in programs.

(3)

1 • Runtime (1)

2 • Syntax (1)

3 • Logic (1)

(f) Here is an algorithm.

```
today = ""

today = int (input ("Day of the week (1-7)"))

if ((today == 6) or (today == 7)):
    print ("8am")
elif ((today >= 1) and (today <= 3)):
    print ("7:30am")
else:
    print ("7am")
```

Complete the table to show the output for the given input.

(3)

One mark for each correct cell

Input	Output
2	7:30am
7	8am
8	7am

(Total for Question 1 = 17 marks)

S 7 0 4 6 1 A 0 5 1 6

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

4 Computational thinking

(a) There are different types of sorting and searching algorithms.

(i) Name the sort algorithm where the next highest number always moves into the correct position on each pass of a loop.

(1)

Award **one** mark for any of the following:

- Bubble (1)
- Bubble sort (1)

(ii) Name the search algorithm that may have to look at all the items in an array.

(1)

Award **one** mark for any of the following:

- Linear (1)
- Linear search (1)

(b) Programmers use a variety of techniques to write algorithms.

(i) Describe the selection construct used in algorithms.

Award up to **two** marks for a linked description such as: • A selection construct is used to decide (1) between multiple options/choices (1) • A selection construct chooses an execution path (1) based on the outcome of (relational) tests (1) • A selection construct changes the flow of execution/makes decisions (1) by using if/elif/else (1) • A condition is used to determine between two courses of action (1). If the condition evaluates to True a block of code is executed / evaluates to False, the block of code is skipped. (1)	For both marks, the expansion must associate with the statement.

(ii) Describe **one** way that a subprogram is an abstraction.

(2)

Award up to **two** marks for a linked description such as: • The way a subprogram works is hidden (1) behind the name of the subprogram (1). • When a subprogram is called (by name) in a program (1), the way it works is hidden (from the caller) (1) • The programmer needs to know what inputs a subprogram needs (1), but not how it is coded (1)	For both marks, the expansion must associate with the statement. Allow alternatives for subprogram: Functions Procedures Routines Allow modifier 'library', e.g. library routines

(c) Programmers use logical and arithmetic operators to write algorithms.

(i) State which logical operator must be evaluated first. **(1)**

- NOT (1)
- not (1)

(ii) Division, modulus and integer division are often confused.

Explain which operator should be used in a binary search algorithm to find the middle position.

Award up to **two** marks for a linked explanation such as: • Integer division should be used (1) AND one of: • because indices in an array/list cannot be fractional/decimal numbers (1) • because it will always result in a whole number (1) • because the result will be rounded down to the nearest whole integer (1)	For both marks, the expansion must associate with the statement.

(d) This algorithm performs a calculation based on the length of a word inputted.

```
word = ""
subTotal = 0
total = 0

word = input ("Enter a word: ")
while (word != ""):
    subTotal = len (word)
    if (subTotal > 5):
        print ("Long word")
        total = total + subTotal
    elif (subTotal == 5):
        print ("Average word")
        total = total - subTotal
    else:
        print ("Short word")
        total = total * subTotal
    word = input ("Enter a word: ")
print (total)

```

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

The inputs are television, table, cup and a blank line.

Complete the trace table showing the execution of the program with these inputs.

You may not need to fill in all the rows in the table.

(4)

word	subTotal	total	Display
""	0	0	

Award **one** mark for each correct row showing relationship of variable changes.

word	subTotal	total	Display
""	0	0	
television	10	10	Long word
table	5	5	Average word
cup	3	15	Short word
""			15

Different versions of trace tables are acceptable

Ignore display of prompts

Allow follow-through on arithmetic errors only between rows

(e) A program is being developed to process information about sweets sold in a shop. All sweets have value-added tax (VAT) of 20% added to their price.

The program will:

- read the data from a text file and store it in the program
- calculate the value of the stock for each sweet name
- calculate the total value of all the sweets
- calculate the amount of VAT due on the total value.

In each row of the text file there is a sweet name, its price and the quantity in stock.

Here is part of the data file.

```
Corn Cremes ,0.83, 77
3 Fruits, 1.22, 55
Nanny Nougat ,0.65,44
Jelly Jacks, 0.59, 33
Toffee-9, 1.30, 22
```

Discuss how the required data should be stored in the program.

Your answer should include:

- which data is stored as variables, constants, strings and records
- reasons for your decisions.

6)

Indicative content:

Variables

- The result of the calculation of the value of the stock for each sweet should be stored in a variable (real). It is calculated by multiplying the wholesale price x the number in stock.
- The total value of all the sweets should be stored in a variable (real). It is calculated by adding up all the individual values for each sweet.
- The amount of value-added tax (VAT) due on the total value of all sweets (real) should be stored in a variable. It is calculated by multiplying 0.20 x the total value of stock.
- Individual items, read from the file, should be stored in variables, such as name (string), wholesale price (real), and number in stock (integer).

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

Constants

- VAT should be stored as a constant.
- This is because its value will not change while the program is running.
- Should the value need to change in the future, it only needs to be changed in one place.
- Saving VAT as a constant, with a meaningful name, will also help programmers make fewer mistakes, because the constant's name will make sense in the context.

Strings

- The sweet names should be stored as strings, because they have uppercase and lowercase letters.
- A string will also store the names that have digits, blanks, or special symbols in them.

Records

- All the information about one sweet should be stored in a record, so that it can be easily accessed together.
- A record is made up of fields of data, which may be of different data types.
- A record for a sweet contains a string (name), a real number (wholesale price), and an integer (number in stock).

Level	Mark	Descriptor
	0	No rewardable content.
Level 1	1–2	Basic, independent points are made, showing elements of understanding of key concepts/principles of computer science. (AO1) The discussion will contain basic information with little linkage between points made or application to the context. (AO2)
Level 2	3–4	Demonstrates adequate understanding of key concepts/principles of computer science. (AO1) The discussion shows some linkages and lines of reasoning with some structure and application to the context. (AO2)
Level 3	5–6	Demonstrates comprehensive understanding of key concepts/principles of computer science to support the discussion being presented. (AO1) The discussion is well developed, with sustained lines of reasoning that are coherent and logically structured, and which clearly apply to the context. (AO2)

5 Computational thinking

(a) State the type of error that can be found in **algorithms**.

(1)

Logic

(b) A binary search algorithm is used with this list to find the target value 'b'.

a b c d e f g h i j k

Complete the table to show the **three** characters in the order that the algorithm would compare them against the target value.

(3)

First	f
Second	c
Third	a
Fourth	b

(c) Programmers can use all capitals to show that a value is a constant.

A constant is shown here on line 3.

```
2  # Prototype for the main swimming pool
3  MAX_CAPACITY = 120        # Maximum number of swimmers
4
5  numAdult = 14             # Current number of adults swimming
6  numChild = 73             # Current number of children swimming
```

Explain **one** reason why programmers use signals indicating a value is a constant, rather than repeating the same fixed value throughout an algorithm.

(2)

Award **1** mark for the identification of a reason (1) with a linked justification/exemplification (1), up to a maximum of **2** marks.

- Constants (shown in all capitals) are less likely to be changed by accident or error (1), so algorithms that use them should be more robust (1).
- If the value of a constant does have to be altered (1), only one change is required (on the line where it is created and set) (1).
- Constants allow values to be replaced with a name/identifier (1), so code is easier to read/maintain (1).

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA

(d) Margaret owns an ice-cream shop.

This program manipulates sales figures from Margaret's shop.

```
num = 0
x = 999
y = 0
line = ""

f = open("SalesFile.txt", "r")
for line in f:
    num = int(line)
    if num < x:
        x = num
    if num > y:
        y = num
print(x, y)
f.close()
```

The only inputs from the file to the program are 355, 554, 199 and 409.

Complete the trace table showing the execution of the program with these four inputs.

You may not need to fill in all the rows in the table.

(6)

num	x	y	Display

1 mark for initialising all variables and **1** mark for each correct pass through the loop.

num	x	y	Display	Marks
0	999	0		(1)
355	355	355		(1)
554		554		(1)
199	199			(1)
409				(1)
			199 554	(1)

- Award alternative versions of the trace table if correct. For example, copying of values that do not change.
- Passes are incorrect if display is indicated.
- Display must be after the final pass (on a separate line in the table).

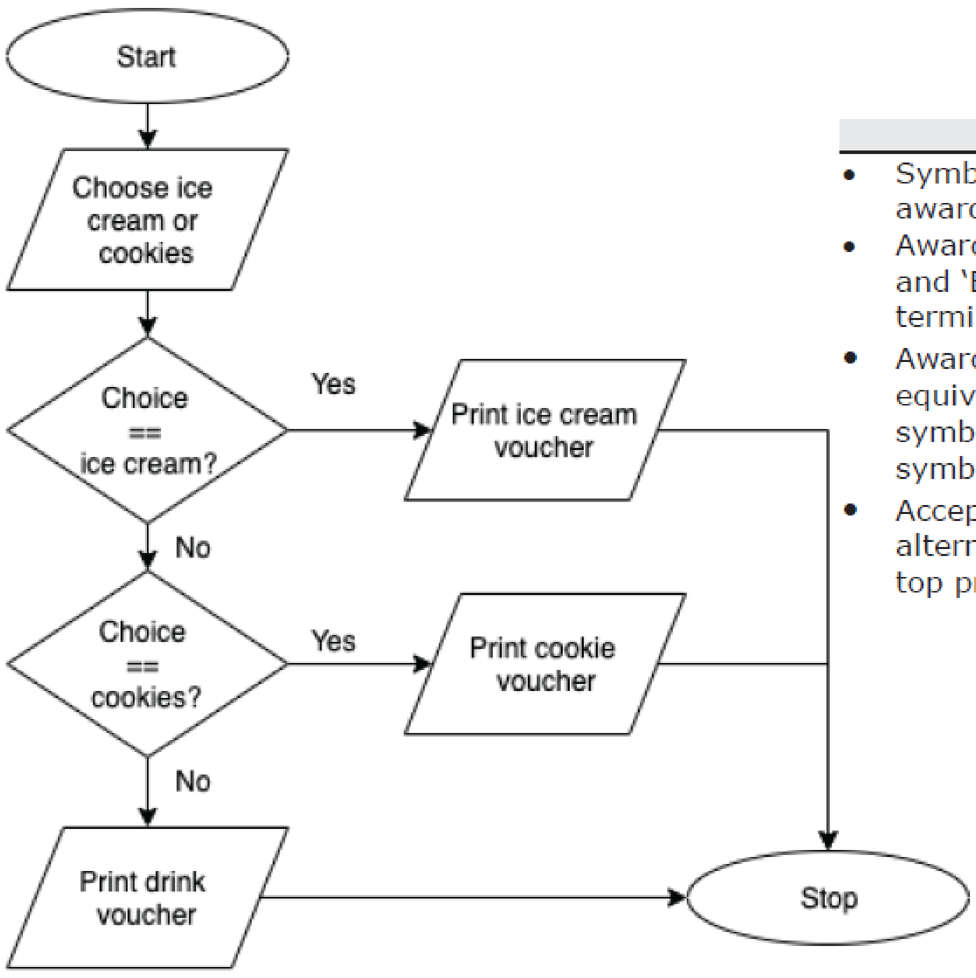
(e) Customers can get a voucher for their favourite item.

- Customers whose favourite item is ice cream get a voucher for ice cream.
- Customers whose favourite item is cookies get a voucher for cookies.
- Customers who do not choose either get a voucher for drinks.

Complete the flowchart to show this process.

(6)

- Correct message in output box acting as a prompt for the user (1).
- Correct diamond symbol for decision (1).
- Correct test 'Choice == cookies?' for decision (1).
- Correct label 'Yes' on right arrow AND Correct label 'No' on bottom arrow (1).
- Correct output symbol with suitable message (1).
- Correct ellipse symbol and 'stop' for terminator (1).



- Symbol and contents are awarded independently.
- Award 'End', 'Stop', 'Start' and 'Begin' as text for terminator symbols.
- Award '==' and '=' used for equivalence inside decision symbol, but not in process symbol.
- Accept 'Input choice' as an alternative response in the top process symbol

(Total for Question 5 = 18 marks)

TOTAL FOR PAPER = 75 MARKS

DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA DO NOT WRITE IN THIS AREA