CAMBRIDGE IGCSE (0478)

Module K: Algorithm design

Workbooklet mark scheme · 9 parts · 190 marks

Contents

Part	Topic	Marks
K1	Sub-systems, abstraction, decomposition, and inputs, processes, outputs and storage: mark scheme	21
K3	Structure diagrams: mark scheme	21
K4	The program development life cycle: mark scheme	21
K5	Reading and writing pseudocode: mark scheme	20
K6	Flowcharts: mark scheme	22
K7	Turning a problem into an algorithm: mark scheme	22
K8	Trace tables and dry-runs: mark scheme	21
K9	Finding and fixing errors in an algorithm: mark scheme	21
K10	Explaining the purpose of an algorithm: mark scheme	21
	Total	**190**

MODULE K ALGORITHM DESIGN · K1

Sub-systems, abstraction, decomposition, and inputs, processes, outputs and storage: mark scheme

21 marks · Cambridge 0478 syllabus 7.1 (analysis: abstraction, decomposition of the problem), 7.2(a) (every computer system is made up of sub-systems, which are made up of further sub-systems) and 7.2(b) (how a problem can be decomposed into its component parts: inputs, processes, outputs, storage)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding. In questions 2, 3 and 5, answers must apply to the scenario given.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.2(a)	B (Every computer system is made up of sub-systems, which are made up of further sub-systems)	**1**	More than one box ticked scores 0.
2(a) 0478 7.2(a)	Any two from: Temperature sensing // reading the room temperatures User interface // phone app // owner's settings Timetable // scheduling // timer Radiator control // switching radiators on and off // heating control Comparing room temperature with target temperature // decision making	**2**	max 2. Credit any other sensible sub-system of this heating system. Accept a named hardware or software part with a clear job (e.g. "the sensors that read the temperature"). Do not accept "input" / "output" / "hardware" / "software" alone.
2(b) 0478 7.2(a)	Two further sub-systems that belong to the chosen sub-system, e.g.: **Phone app:** set the target temperature // set the weekly timetable // display the current temperature of each room // log in **Temperature sensing:** read a sensor in each room // convert / send the reading to the controller **Timetable:** store the timetable // check the current day and time against the timetable **Radiator control:** decide if a radiator should be on // send the on / off signal to each radiator	**2**	One mark per valid further sub-system, max 2. Each must be a smaller part of the sub-system chosen. If no sub-system is named, credit answers that are clearly smaller parts of one sub-system from 2(a). Do not credit a repeat of a sub-system already given in 2(a) as a separate sub-system.
3(a) 0478 7.1 (analysis)	Any two from: Removing / discarding / ignoring details (of the problem) that are not relevant // not needed // hiding complexity Keeping // focusing on the key elements // the details needed to solve the problem Simplifying the problem	**2**	max 2. Do not accept "breaking the problem into parts" (decomposition). No mark for repeating that it happens at the analysis stage.

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) 0478 7.1 (analysis)	Any two from: Engine room // machinery // pipes // wiring Crew-only areas // staff rooms Exact sizes / measurements of rooms // decks Furniture // fittings inside rooms Materials // colours of walls / floors The sea // the shape of the hull // the ship's route	**2**	max 2. Credit any other detail a passenger does not need to find their cabin, the café, the toilets or the lifeboat stations. Do not credit cabins, café, toilets, lifeboat stations, deck numbers, stairs / lifts or "you are here".
4(a) 0478 7.2(b)	Number of steps shown on the watch face: Output Heart rate read from a sensor: Input Step totals for the last 30 days: Storage Working out the average heart rate: Process	**4**	One mark per correct row. A row with more than one box ticked scores 0.
5(a) 0478 7.2(b)	**Inputs** (max 2): Membership number // card ID (from scanning the card) Maximum number of people allowed in the pool (entered by the lifeguard) Today's date // the current date and time (from the system clock) **Processes** (max 2): Look up / search for the member's details Check the membership has not expired // compare the expiry date with today's date Check whether the pool is full // compare the number in the pool with the maximum Add 1 to the number of people in the pool **Outputs** (max 2): (Signal to) open the gate Welcome message with the member's name Message saying why the member cannot enter // membership expired // pool full **Storage** (max 2): Members' details // membership numbers, names, expiry dates Date and time of every entry Number of people in the pool Maximum number of people allowed	**8**	One mark per correct entry in the correct row, max 2 per row. An answer in the wrong row scores 0 (e.g. "welcome message" as storage). The same answer written twice scores once. The maximum number of people can be credited as both an input and storage. Accept "scan the card" as an input.

Total: 21 marks

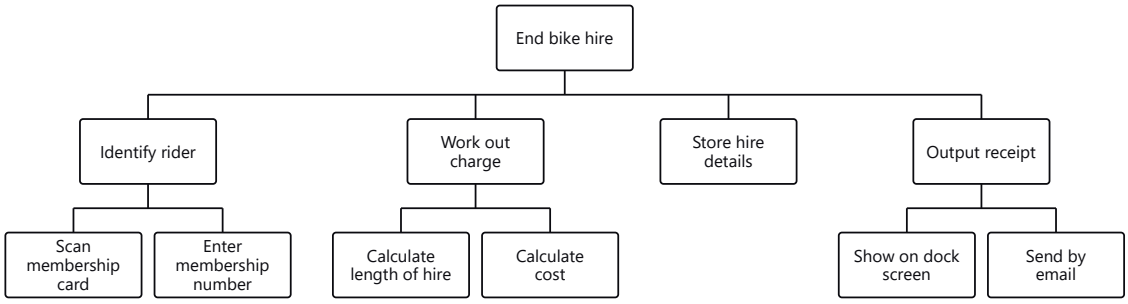
MODULE K ALGORITHM DESIGN · K3

Structure diagrams: mark scheme

21 marks · Cambridge 0478 syllabus 7.1 (design stage: structure diagrams), 7.2(a) (every computer system is made up of sub-systems, which are made up of further sub-systems), 7.2(b) (decomposing a problem into inputs, processes, outputs and storage), 7.2(c) (structure diagrams as a method to design and construct a solution)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.2(a), 7.2(c)	Shows how a system is broken down into sub-systems: True Top box represents the whole system: True Uses the decision symbol: False A sub-system can be broken down into further sub-systems: True	**4**	One mark per correct row. A row with both boxes ticked scores 0. (The decision symbol is a flowchart symbol.)
2(a) 0478 7.1	Design	**1**	Do not accept analysis / coding / testing.
2(b) 0478 7.2(a), 7.2(c)	Any two from: How a system / problem is broken down / decomposed into sub-systems // the component parts of a problem The sub-systems arranged in levels / a hierarchy, with the whole system at the top How each sub-system is broken down into further / smaller sub-systems on the level below How the sub-systems are linked // which sub-system each smaller part belongs to	**2**	max 2. Credit any other valid answer. Do not accept "the order of the steps" / "the decisions" / "the flow of the algorithm" (these describe a flowchart).
3(a) 0478 7.2(a)	Decide action	**1**	Do not accept "Greenhouse system".
3(b) 0478 7.2(b)	Read soil moisture sensor: Input Compare readings with target values: Process Switch water pump on or off: Output Store target values: Storage	**4**	One mark per correct row. A row with more than one box ticked scores 0.
3(c) 0478 7.2(c)	Add a new box / sub-system for sending the text message // alert Join it with a line below Control devices // join it with a line below Greenhouse system as a new (output) sub-system	**2**	One mark per bullet. Second mark: placed as an output. Do not accept it below Take readings or Keep records. An extra box to check if it is too hot may also be added.

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) 0478 7.2(b), 7.2(c)	One mark for a hierarchical structure: sub-systems joined by lines below End bike hire, with a further level below at least one of them One mark for suitable names for the four sub-systems: identify rider, work out charge, store hire details, output receipt One mark for sub-systems for scanning the membership card **and** entering the membership number (below identify rider) One mark for sub-systems for calculating the length of the hire **and** calculating the cost (below work out charge) One mark for sub-systems for showing the receipt on the dock screen **and** sending it by email (below output receipt)	**5**	One mark per bullet. Accept Input / Process / Storage / Output as the names of the four sub-systems. Store hire details may have further sub-systems below it. Sub-systems may be in any order within a level. A flowchart (steps in sequence, decision symbols) is not a structure diagram: 0 marks.
4(a) Example answer			
5(a) 0478 7.2(a)	One mark for: the system is divided into smaller sub-systems, each doing one part of the task, with an example // e.g. ending a hire is made up of identifying the rider, working out the charge, storing the details and outputting a receipt One mark for: each sub-system can itself be divided into further / smaller sub-systems, with an example // e.g. identifying the rider is made up of scanning a card and entering a membership number	**2**	max 1 if no example from the bike hire system is given. Accept examples from the candidate's own diagram in Question 4 if they match the description.

Total: 21 marks

MODULE K ALGORITHM DESIGN · K4

The program development life cycle: mark scheme

21 marks · Cambridge 0478 syllabus (exams 2026-2028) 7.1 (understand the program development life cycle, limited to analysis, design, coding and testing) and 7.2(c) (methods to design a solution: structure diagrams, flowcharts, pseudocode)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding. The syllabus limits the life cycle to four stages: do not credit other stage names (e.g. maintenance, evaluation, implementation).

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.1	One mark for each correct stage in the correct position: Stage 2: Design Stage 3: Coding Stage 4: Testing	**3**	Accept "programming" // "writing (the) code" for Coding. Do not credit "implementation". A correct stage in the wrong box scores 0 for that box.
2(a) 0478 7.1	One mark for each correct row: Asking the centre manager what the program must do: Analysis Writing the program code for the cost module: Coding Drawing a structure diagram: Design Running the finished program with a group size of 9: Testing Deciding which details the program does not need: Analysis	**5**	A row with more than one box ticked scores 0. Row 5 is abstraction, which the syllabus places in analysis.
3(a) 0478 7.1	Any two from: Book a session of 1, 2 or 3 hours // input / store the session length Accept / check / validate a group size of 1 to 8 climbers Check that every climber has completed a safety induction (before booking them in) Output a booking reference Output the total cost (of the session)	**2**	max 2. The last note gives two requirements: credit the booking reference and the total cost separately. Do not credit any of the last three notes (walls, café, opening year). Credit any other valid requirement drawn from the notes.
3(b) 0478 7.1	Any two from: The walls were repainted // the colour of the walls The café (sells hot drinks) The year the centre opened // 2015	**2**	max 2. Do not credit any requirement listed in 3(a): those are needed by the program.

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) 0478 7.1, 7.2(c)	One mark for each empty box containing a different valid sub-problem, e.g.: Enter / input the booking details // the session length // the number of climbers Check / validate the group size (1 to 8) // the session length (1, 2 or 3 hours) Calculate / work out the total cost Create / generate the booking reference Output the booking reference // the total cost // the booking details Store / save the booking	**2**	Credit any other valid sub-problem of the booking program. Do not credit: a repeat of the given box; the same sub-problem in both boxes; the whole problem ("book a session"); an item that is not part of the program (e.g. the walls or the café).
4(b) 0478 7.1, 7.2(c)	Flowchart(s) Pseudocode	**2**	Do not credit "structure diagram" (already used), "program code" or "trace table".
5(a) 0478 7.1	Any two from: Each module / part of the program is tested as (soon as) it is written // before the next module is written Modules are tested separately / individually (e.g. the cost module on its own) Any errors found are corrected // the code is changed ... and the module is tested again // testing is repeated until the module works (as expected)	**2**	max 2. Do not credit "testing the program" or "checking it works" alone: the answer must show testing is repeated / carried out during the writing of each part.
6(a) 0478 7.1 (7.6)	Any three from: Choose / create test data (for the program) // write a test plan ... using normal, abnormal, extreme and boundary data // a correct example in context (e.g. a group size of 4 as normal data; 1 or 8 as extreme data; 1 and 0 // 8 and 9 as boundary data (1 / 8 accepted, 0 / 9 rejected); 12 or "ten" as abnormal data) Work out the expected result / outcome for each test Run the (complete) program with the test data Compare the actual results with the expected results // record the results If a result is not as expected, find and correct the error (and test again) Check that the program meets (all) the requirements (from the analysis stage)	**3**	max 3. Credit an example only if everything the candidate gives is correct: any type of test data named must match the value, and any expected outcome must be correct (a group size of 1 to 8 is accepted; 0, 9 or a non-integer is rejected). Do not credit an example with the wrong type of test data (e.g. "9 as normal data"). Credit any other valid test-data example from the scenario (e.g. a session length). Do not credit "check it works" or "look for errors" alone.

Total: 21 marks

MODULE K ALGORITHM DESIGN · K5

Reading and writing pseudocode: mark scheme

20 marks · Cambridge 0478 syllabus 7.2(c) (design methods: pseudocode), 7.3 (purpose of a given algorithm), 7.8 (identify and correct errors), 7.9 (write algorithms in pseudocode; precision), 8.1.1 (declare variables and constants), 8.1.4 (selection, iteration, counting, totalling, operators) and section 4 (pseudocode)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. **...** at the start of a point means it builds on the point before. Code answers must be in pseudocode; ignore minor differences in capitals, spacing and indentation. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.2(c)	Row 2 ticked only (keywords and structured, code-like statements without the rules of one programming language)	**1**	More than one tick = 0. Row 1 is a structure diagram, row 3 a flowchart, row 4 a trace table.

2(a) 0478 7.9, 8.1.4(f)	One mark for each correct condition:
	`Distance > 100`
	`Answer <> 'N'`
	`Hours >= 1 AND Hours <= 12` // `(Hours >= 1) AND (Hours <= 12)` // `Hours > 0 AND Hours < 13`
		3	Allow `IF` in front of the condition. Row 2: allow `"N"`; allow `NOT(Answer = 'N')`; N must be in quotes; do not accept `!=` (not a pseudocode operator). Row 3: do not accept `Hours >= 1 AND <= 12` (second comparison has no identifier), `OR` instead of AND, `1 <= Hours <= 12`, or => / ≥ / ≤.

3(a) 0478 8.1.1, section 4	One mark for each correct statement:
	`CONSTANT MaxSeats ← 52`
	`DECLARE SeatsLeft : INTEGER`
		2	Allow = or <– for ← in the constant. Do not accept `REAL` or `STRING` for the variable. Do not accept a variable declaration for MaxSeats.

4(a) 0478 7.3	Any two from: To input the weights of 5 loaves To total / add up the weights (of the loaves) To count how many loaves weigh 400 (grams) or more // are accepted To output the total weight and the number of loaves accepted	**2**	max 2. Do not accept "more than 400" / "over 400" for the counting point (400 is accepted). Do not accept "checks the loaves" alone.
4(b)(i) 0478 8.1.4(c)	07	**1**	Allow 7.

QUESTION	ANSWER	MARKS	GUIDANCE
4(b)(ii) 0478 8.1.4(d)	13	**1**	Correct answer only. Line 10 is totalling, not counting.

4(c)
0478 7.3
	One mark for each correct output:
	`Total weight: 2013`
	`Loaves accepted: 3`
		2	Ignore quotation marks. Allow 2013 and 3 without the message text if each is clearly the correct output.

5(a) 0478 7.8	One mark for each error identified and correctly corrected: Line 05: `FOR Day ← 1 TO 7` Line 08: `IF Visitors > Highest` Line 10: `Highest ← Visitors` Line 13: `OUTPUT "Highest number of visitors: ", Highest`	**4**	A correction scores only with its correct line number. Line 05: allow `FOR Day ← 0 TO 6` (also 7 days). Line 08: allow >=. Allow a clear description, e.g. "change < to >", "store Visitors not Day in Highest", "output Highest not Visitors".

6(a)
0478 7.9, 8.1.4(c)(d)
	Any four from:
	Count initialised to 0 before the loop
	Condition-controlled loop that stops when −1 is input // `REPEAT ... UNTIL Time = -1` // `WHILE Time <> -1 DO ... ENDWHILE`
	Input of a journey time inside the loop
	`IF Time > 30` … `THEN Count ← Count + 1` … `ENDIF`
	Output of the count with a message, after the loop
	`DECLARE Time : INTEGER`
	`DECLARE Count : INTEGER`
	`Count ← 0`
	`OUTPUT "Enter journey time in minutes, -1 to stop"`
	`INPUT Time`
	`WHILE Time <> -1 DO`
	`  IF Time > 30`
	`  THEN`
	`  Count ← Count + 1`
	`  ENDIF`
	`  OUTPUT "Enter journey time in minutes, -1 to stop"`
	`  INPUT Time`
	`ENDWHILE`
	`OUTPUT "Journeys over 30 minutes: ", Count`
		4	max 4. A WHILE loop gets the loop mark if Time has a value before the condition is first tested: either an input before the loop and again at the end of the loop body, or Time set to a value other than −1 before the loop and the input at the start of the loop body. A REPEAT ... UNTIL loop is also fine (−1 is not more than 30, so it is never counted). Do not award the selection mark for `>= 30`. Do not award the output mark if the count is output inside the loop. A FOR loop does not get the loop mark. Program code (e.g. Python) scores 0.

Total: 20 marks

MODULE K ALGORITHM DESIGN · K6

Flowcharts: mark scheme

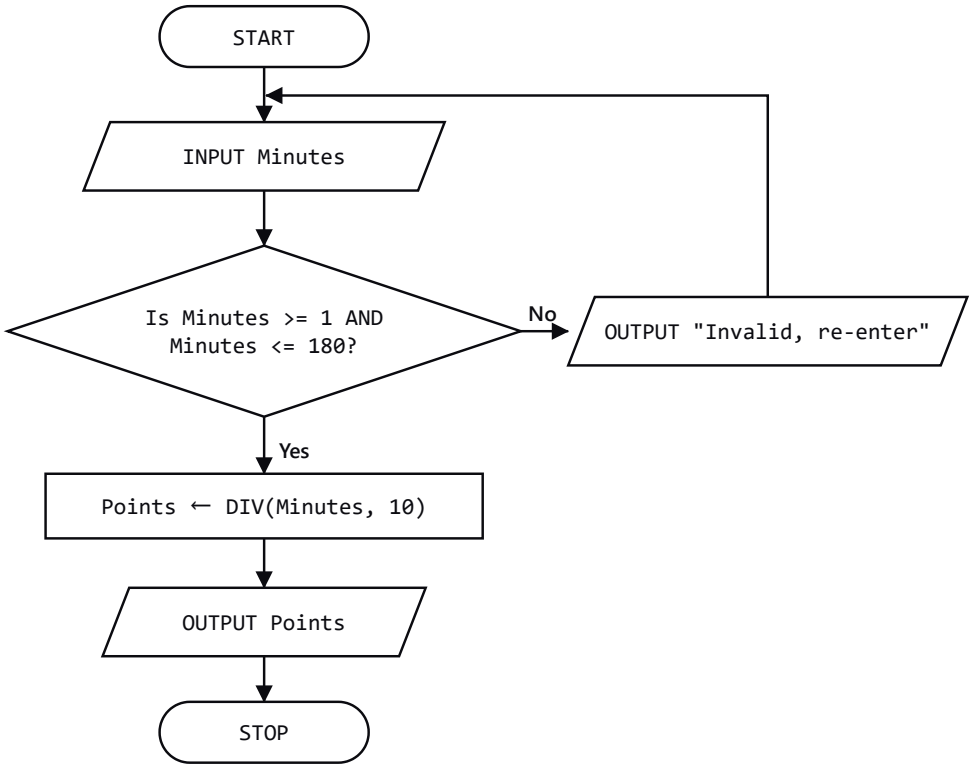
22 marks · Cambridge 0478 syllabus 7.3 (purpose of an algorithm), 7.7 (trace tables), 7.8 (identify and correct errors), 7.9 (write and amend algorithms using flowcharts) and section 4 (flowchart symbols)

How to use. One mark per bullet point up to the max shown. **//** separates alternative answers; **/** separates alternative words. Words in (brackets) are not needed for the mark. Credit any other valid answer that shows the same understanding. Conditions must be written precisely (e.g. `Minutes > 180`, not "Minutes is more than 180").

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
0478 section 4</td><td>Decision (diamond) → question with two possible outcomes, such as Yes or No
Terminator (rounded rectangle) → where the algorithm starts or stops
Subroutine (rectangle with double sides) → call to a separate flowchart that is not joined to this one
Input/Output (parallelogram) → data going into, or coming out of, the algorithm</td><td>4</td><td>One mark for each correct line. More than one line from a symbol scores 0 for that symbol. The description "something being performed or done" (process) is not used.</td></tr>
<tr><td>2(a)
0478 7.7</td><td>Count: 0, 1, 2, 3, 4, 5
Cars: 38, 45, 22, 45, 41
Busy: 0, 1, 2, 3
Largest: 0, 38, 45
OUTPUT: 3, 45
<table><tr><th>COUNT</th><th>CARS</th><th>BUSY</th><th>LARGEST</th><th>OUTPUT</th></tr><tr><td>0</td><td></td><td>0</td><td>0</td><td></td></tr><tr><td></td><td>38</td><td></td><td>38</td><td></td></tr><tr><td>1</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>45</td><td>1</td><td>45</td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>22</td><td></td><td></td><td></td></tr><tr><td>3</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>45</td><td>2</td><td></td><td></td></tr><tr><td>4</td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>41</td><td>3</td><td></td><td></td></tr><tr><td>5</td><td></td><td></td><td></td><td>3</td></tr><tr><td></td><td></td><td></td><td></td><td>45</td></tr></table></td><td>5</td><td>One mark per correct column (values in the correct order). Values may sit on different rows from those shown. Largest must not change for the second 45. OUTPUT: 3 then 45, with no other values.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
2(b) 0478 7.3	Any two from: inputs the number of cars for each of five crossings counts the number of crossings with more than 40 cars // counts the busy crossings finds the largest number of cars carried on a crossing outputs the number of busy crossings and the largest number of cars	**2**	max 2. Do not accept "counts the cars" alone or "finds the total". Do not accept "40 or more" (the check is more than 40).
3(a) 0478 7.9 (totalling 7.4)	One mark per bullet: set a total to zero before the loop, e.g. add `Total ← 0` to the first process box // before `INPUT Cars` add a process box inside the loop, after `INPUT Cars`, with `Total ← Total + Cars` add an output box after the loop, e.g. `OUTPUT Total` on the Yes branch of `Is Count = 5?` (before STOP)	**3**	Any sensible variable name. The first mark is not given if the total is set to zero inside the loop. The second mark needs the addition to be inside the loop. Accept the changes described in words if the position is clear.
4(a) 0478 7.8	One mark for each error identified **and** corrected: `INPUT Speed` is in a process symbol // rectangle → use an input/output symbol // parallelogram `Is Speed < 30?` outputs "Slow down" for slow vehicles → `Is Speed > 30?` // `Is Speed <= 30?` with the Yes/No labels swapped the No flow line from `Is Count = 10?` goes back above `Count ← 0`, so Count is reset each time and the loop never ends → the flow line should go back to `INPUT Speed` // between `Count ← 0` and `INPUT Speed`	**3**	Errors may be given in any order. Do not accept `Is Speed >= 30?` (30 is not over 30). For the third error accept "Count ← 0 is inside the loop" with a correction that moves the loop-back point below it.
5(a) 0478 7.9, section 4 (range check 7.5(a))	One mark per bullet point, max 5: START and STOP terminators used `INPUT Minutes` in an input/output symbol decision symbol checking both limits, e.g. `Is Minutes >= 1 AND Minutes <= 180?` // two decisions // `Is Minutes < 1 OR Minutes > 180?` with Yes/No to match ...if invalid, an error message is output (input/output symbol) and a flow line goes back to the input `Points ← DIV(Minutes, 10)` in a process symbol `OUTPUT Points` in an input/output symbol, after the calculation, before STOP all flow lines have arrows and both decision outcomes are labelled Yes/No (or True/False) *An example answer is drawn below the table.*	**5**	Accept `Minutes DIV 10`. Do not accept `Minutes / 10` (not integer division) or ROUND. Do not award the decision mark for a condition written in words (e.g. "Is it between 1 and 180?") or if only one limit is checked. Accept `Minutes > 0` for the lower limit. Non-standard symbols lose the mark for that box only.

Example answer for 5(a)



Total: 22 marks

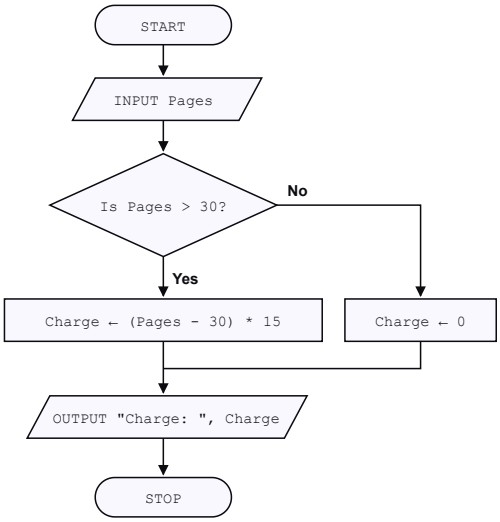
MODULE K ALGORITHM DESIGN · K7

Turning a problem into an algorithm: mark scheme

22 marks · Cambridge 0478 syllabus 7.2(b) (decompose a problem: inputs, processes, outputs, storage), 7.2(c) (design methods: flowcharts, pseudocode), 7.9 (write and amend algorithms using pseudocode and flowcharts), 8.1 (variables; sequence, selection, iteration; totalling and counting)

How to use. One mark per bullet point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Algorithms must be precise: `Pages > 30` is acceptable, "pages is more than thirty" is not. Questions 3 and 4(b) must be answered in pseudocode; program code scores 0. Ignore minor syntax slips. Example answers are one possible response, not the only one.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.2(b)	**Input:** the number of pages printed (today) **Process:** any one from: check whether more than 30 pages were printed // subtract 30 from the number of pages // multiply the extra pages by 15 // calculate the charge **Output:** the charge (in cents) **Storage:** any one from: the number of pages // the charge // the number of free pages (30) // the cost per page (15 cents)	**4**	One mark per row. Credit any other valid process or stored item. Do not accept "the charge" as the input, or "the number of pages" as the output. Do not accept "printing" as the process, or "the printer" / "a database" alone as storage.
1(b) 0478 8.1.4(a)–(c)	Multiply the extra pages by 15: **Sequence** Charge nothing if 30 pages or fewer: **Selection** Keep asking until 1 to 500 is entered: **Iteration**	**3**	One mark per correct row. A row with more than one tick scores 0.

QUESTION	ANSWER	MARKS	GUIDANCE
2 0478 7.9, 7.2(c)	One mark per bullet point. Max 5. START and STOP terminators Input of the number of pages in an input/output symbol Decision symbol checking more than 30 pages (e.g. `Is Pages > 30?`) with both exits labelled Yes/No (True/False) the right way round Charge calculated as (Pages − 30) * 15 on the "more than 30" route Charge set to 0 on the "30 or fewer" route // charge set to 0 before the decision Output of the charge in an input/output symbol Flow lines with arrows joining every symbol, with both routes reaching the output and STOP Example answer:  	**5**	Max 5. Accept `Pages * 15 - 450` for the calculation. Accept a decision such as `Is Pages <= 30?` with the Yes/No routes the other way round. Accept `Is Pages >= 30?` (or `Is Pages < 30?` with the Yes/No routes swapped), because at exactly 30 pages (30 − 30) * 15 = 0, so the charge is still correct. Do not award the decision mark if the exits are not labelled. Non-standard symbols (e.g. a rectangle for an input): do not award that symbol's mark. Do not accept a decision written in vague English ("Is it over the limit?").

QUESTION	ANSWER	MARKS	GUIDANCE

3(a)
0478 7.9
	One mark per bullet point. Max 3.
	Input of the number of pages
	A condition-controlled loop (REPEAT … UNTIL or WHILE … DO … ENDWHILE) with the correct closing keyword
	… with a correct condition that checks both limits (e.g. `UNTIL Pages >= 1 AND Pages <= 500` // `WHILE Pages < 1 OR Pages > 500 DO`)
	The number of pages is input again inside the loop
	Example answers:
	`REPEAT`
	`  OUTPUT "Enter the number of pages (1 to 500)"`
	`  INPUT Pages`
	`UNTIL Pages >= 1 AND Pages <= 500`
	//
	`INPUT Pages`
	`WHILE Pages < 1 OR Pages > 500 DO`
	`  OUTPUT "Invalid, enter a number from 1 to 500"`
	`  INPUT Pages`
	`ENDWHILE`
		3	Max 3. An IF statement with no loop (checks only once): max 1. Do not credit a condition with a missing variable (e.g. `Pages < 1 OR > 500`) or AND / OR the wrong way round. Accept `Pages <= 0 OR Pages >= 501`. Do not accept "between 1 and 500" in English.

4(a) 0478 8.1.1	Total: a meaningful identifier, e.g. `TotalPassengers` // `Total`, **and** initial value 0 Count: a meaningful identifier, e.g. `BusyCrossings` // `Count`, **and** initial value 0	**2**	One mark per row; both the identifier and the initial value are needed. Do not accept identifiers containing spaces, single letters, or keywords (e.g. `INPUT`, `OUTPUT`). Do not penalise the use of lower case or underscores.

QUESTION	ANSWER	MARKS	GUIDANCE

4(b)
0478 7.9, 8.1.4(c)(d)
	One mark per bullet point. Max 5.
	A loop that repeats 8 times, with the correct closing keyword (e.g. `FOR Crossing ← 1 TO 8` … `NEXT Crossing`)
	Input of the number of passengers inside the loop
	Totalling inside the loop (e.g. `Total ← Total + Passengers`)
	IF statement checking `Passengers > 150`, with ENDIF … counting only when the condition is true (e.g. `Count ← Count + 1`)
	Total **and** count both output after the loop, with suitable messages
	Example answer:
	`TotalPassengers ← 0`
	`BusyCrossings ← 0`
	`FOR Crossing ← 1 TO 8`
	`  OUTPUT "Enter the number of passengers on this crossing"`
	`  INPUT Passengers`
	`  TotalPassengers ← TotalPassengers + Passengers`
	`  IF Passengers > 150`
	`  THEN`
	`  BusyCrossings ← BusyCrossings + 1`
	`  ENDIF`
	`NEXT Crossing`
	`OUTPUT "Total passengers today: ", TotalPassengers`
	`OUTPUT "Crossings with more than 150 passengers: ", BusyCrossings`
		5	Max 5. Pseudocode only; program code scores 0. Accept a REPEAT or WHILE loop with a counter that correctly runs 8 times. Accept `>= 151`; do not accept `>= 150`. Accept = or <- for assignment. Do not award the totalling / counting mark if the total or count is reset to 0 inside the loop. Outputs inside the loop do not gain the output mark.

Total: 22 marks

MODULE K ALGORITHM DESIGN · K8

Trace tables and dry-runs: mark scheme

21 marks · Cambridge 0478 syllabus 7.7: complete a trace table to document a dry-run of an algorithm (variables, outputs and user prompts at each step); 7.3: state the purpose of an algorithm; 7.8: identify errors and suggest corrections

How to use. One mark per bullet point / per correct column, up to the max shown. **//** separates alternative answers. Words in (brackets) are not needed for the mark. **Trace tables:** a column is correct when its values appear in the right order; repeated values on later rows are not penalised, and rows may be laid out differently from the example. Outputs are copied exactly but without quotation marks; ignore small spacing slips and a missing colon. A column that carries on after the stopping condition has been met is not correct. Credit any other valid answer that shows the same understanding.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
0478 7.7</td><td>Working through the algorithm by hand, one step at a time, recording what happens (second row)</td><td>1</td><td>Two or more ticks = 0.</td></tr>
<tr><td>1(b)
0478 7.7</td><td>Any one from:
• outputs // the values output
• user prompts // messages asking the user for input</td><td>1</td><td>Do not accept “inputs” alone, “line numbers”, or “errors”.</td></tr>
<tr><td>2(a)
0478 7.7</td><td>One mark for each correct column:
Total: 0, 18, 33, 53
Count: 0, 1, 2, 3
Seats: 18, 25, 15, 20 (12 is not used)
OUTPUT: Enter seats, Enter seats, Too many, Enter seats, Enter seats, Bookings accepted: 3
<table>
<tr><th>Total</th><th>Count</th><th>Seats</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>0</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>Enter seats</td></tr>
<tr><td>18</td><td>1</td><td>18</td><td></td></tr>
<tr><td></td><td></td><td></td><td>Enter seats</td></tr>
<tr><td></td><td></td><td>25</td><td>Too many</td></tr>
<tr><td></td><td></td><td></td><td>Enter seats</td></tr>
<tr><td>33</td><td>2</td><td>15</td><td></td></tr>
<tr><td></td><td></td><td></td><td>Enter seats</td></tr>
<tr><td>53</td><td>3</td><td>20</td><td></td></tr>
<tr><td></td><td></td><td></td><td>Bookings accepted: 3</td></tr>
</table></td><td>4</td><td>The prompt and the input may share a row. 20 is not greater than 20, so it is accepted: Total 53 and Count 3. 25 must not be added to Total. Seats column containing 12 = 0 for Seats. OUTPUT must include all four prompts, Too many once, and the final message; Bookings accepted: 3 alone = 0 for OUTPUT.</td></tr>
</table>

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(a)
0478 7.7</td><td>One mark for each correct column:
Index: 1, 2, 3, 4, 5, 6
Low: 38, 31, 29 (only)
Above: 0, 1, 2 (only)
OUTPUT: Lowest: 29 then Above limit: 2
<table><tr><th>Index</th><th>Low</th><th>Above</th><th>OUTPUT</th></tr><tr><td></td><td>38</td><td>0</td><td></td></tr><tr><td>1</td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td>1</td><td></td></tr><tr><td>3</td><td>31</td><td></td><td></td></tr><tr><td>4</td><td></td><td>2</td><td></td></tr><tr><td>5</td><td>29</td><td></td><td></td></tr><tr><td>6</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>Lowest: 29</td></tr><tr><td></td><td></td><td></td><td>Above limit: 2</td></tr></table></td><td>4</td><td>40 is not greater than 40, so Above stays at 2 when Index is 6: Above reaching 3 = 0 for Above. Low must not take 45, 52 or 40. Accept both outputs on one row in the right order. Accept repeated values of Low / Above on rows where they do not change.</td></tr>
<tr><td>3(b)
0478 7.3</td><td>Any two from:
• finds // outputs the lowest // minimum (river-level) reading
• counts how many readings are above 40 // greater than 40 (cm)
• outputs the number of readings above 40</td><td>2</td><td>Max 1 if only one of the two results (lowest reading, count above 40) is given. Do not accept “40 or more” / “at least 40” for the count. Do not accept “sorts the readings” or a line-by-line description with no result.</td></tr>
<tr><td>4(a)
0478 7.7</td><td>One mark for each correct column:
Number: 50, 16, 5, 1
Count: 0, 1, 2, 3
OUTPUT: Enter a number (first) ... 3 (last)
<table><tr><th>Number</th><th>Count</th><th>OUTPUT</th></tr><tr><td></td><td></td><td>Enter a number</td></tr><tr><td>50</td><td>0</td><td></td></tr><tr><td>16</td><td>1</td><td></td></tr><tr><td>5</td><td>2</td><td></td></tr><tr><td>1</td><td>3</td><td></td></tr><tr><td></td><td></td><td>3</td></tr></table></td><td>3</td><td>DIV gives the whole-number part: 16.67 / 16.6 / 5.33 = 0 for Number. OUTPUT without the prompt = 0 for OUTPUT. Count must stop at 3.</td></tr>
<tr><td>4(b)
0478 7.7</td><td>0</td><td>1</td><td>The condition 1 > 1 is false at the first check, so the loop body never runs. Accept “Enter a number, 0”. Do not accept 1.</td></tr>
</table>

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr>
<td>5(a)
0478 7.7</td>
<td>Count: 1, 2, 3, 4 and Score: 12, 9, 15, 6
Total: 0, 12, 0, 9, 0, 15, 0, 6
Average: 1.5 and OUTPUT: Enter score (four times) ...
Average is 1.5
<table>
<tr><th>Count</th><th>Total</th><th>Score</th><th>Average</th><th>OUTPUT</th></tr>
<tr><td>1</td><td>0</td><td></td><td></td><td>Enter score</td></tr>
<tr><td></td><td>12</td><td>12</td><td></td><td></td></tr>
<tr><td>2</td><td>0</td><td></td><td></td><td>Enter score</td></tr>
<tr><td></td><td>9</td><td>9</td><td></td><td></td></tr>
<tr><td>3</td><td>0</td><td></td><td></td><td>Enter score</td></tr>
<tr><td></td><td>15</td><td>15</td><td></td><td></td></tr>
<tr><td>4</td><td>0</td><td></td><td></td><td>Enter score</td></tr>
<tr><td></td><td>6</td><td>6</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>1.5</td><td>Average is 1.5</td></tr>
</table>
</td>
<td>3</td>
<td>One mark per bullet. The trace must follow the algorithm as written: a Total column of 0, 12, 21, 36, 42 (the corrected algorithm) = 0 for Total, and Average 10.5 = 0 for the third mark. FT: Average and final OUTPUT correct for the candidate's final Total ÷ 4.</td>
</tr>
<tr>
<td>5(b)
0478 7.8</td>
<td>Line 02
Move <code>Total ← 0</code> to before the FOR loop // before line 01 // outside the loop</td>
<td>2</td>
<td>One mark for the line, one for the correction. Accept "Total is reset to 0 every time round the loop; it should be set to 0 once, before the loop starts". Do not accept "delete line 02" alone (Total would not be initialised). Do not accept changes to line 05 or line 07.</td>
</tr>
</table>

Total: 21 marks

MODULE K ALGORITHM DESIGN · K9

Finding and fixing errors in an algorithm: mark scheme

21 marks · Cambridge 0478 syllabus 7.8 (identify errors in given algorithms and suggest ways of correcting these errors) and 7.7 (complete a trace table to document a dry-run of an algorithm)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Corrections should be in pseudocode; ignore minor slips (e.g. <− or = for ←, missing `THEN`) as long as the logic is clear. Credit any other valid answer that shows the same understanding.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
0478 7.8</td><td>One mark for each error identified and corrected:
Line 06 // <code>HighCount ← 1</code> should be <code>HighCount ← 0</code>
Line 07 // <code>FOR Month ← 1 TO 11</code> should be <code>FOR Month ← 1 TO 12</code>
Line 11 // <code>IF Reading > 400.0</code> should be <code>IF Reading >= 400.0</code>
Line 17 // <code>Total / HighCount</code> should be <code>Total / 12</code></td><td>4</td><td>max 4. The line number alone, or the correction alone with no way of telling which line it replaces, scores 0.
Line 07: accept <code>0 TO 11</code> // <code>2 TO 13</code> (loop runs 12 times).
Line 11: accept <code>>= 400</code> // <code>400.0 <= Reading</code>. Do not accept <code>> 399</code> or <code>> 399.9</code> (readings are REAL).
Line 17: accept <code>Total / 12.0</code>. Do not accept <code>Total / Month</code>.</td></tr>
<tr><td>2(a)
0478 7.7, 7.8</td><td>One mark per correct column:
Runner: 1, 2, 3, 4, 5
Time: 62, 58, 71, 55, 60
Fastest: 0 (only, entered once at the start)
OUTPUT: Fastest time: 0
<table>
<tr><th>RUNNER</th><th>TIME</th><th>FASTEST</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td></td></tr>
<tr><td>1</td><td>62</td><td></td><td></td></tr>
<tr><td>2</td><td>58</td><td></td><td></td></tr>
<tr><td>3</td><td>71</td><td></td><td></td></tr>
<tr><td>4</td><td>55</td><td></td><td></td></tr>
<tr><td>5</td><td>60</td><td></td><td>Fastest time: 0</td></tr>
</table></td><td>4</td><td>Allow Fastest 0 repeated on every row. Do not credit the Fastest column if any other value appears in it.
OUTPUT: allow 0 alone. Ignore quotation marks. Do not penalise a final 6 in the Runner column.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
2(b)(i) 0478 7.8	Any one from: Fastest starts at 0 and no time is less than 0 // the condition `Time < Fastest` is never TRUE So Fastest is never changed / updated (and 0 is output)	**1**	Do not accept "the IF statement is wrong" on its own.
2(b)(ii) 0478 7.8	`Fastest ← 999` // Fastest set to any value larger than any possible time	**1**	Accept any value clearly larger than any possible time, e.g. 999, 9999. Do not accept 0, a negative value, or a value that a real 10 km time could exceed (e.g. 60, 71): the correction must work for any input data, not just this data.
2(b)(iii) 0478 7.7	Fastest time: 55	**1**	Accept 55 alone.
3(a) 0478 7.8	Any one from: The algorithm asks for the number of seats again // the loop repeats 5 is rejected even though it is a valid value	**1**	Do not accept "5 is accepted" or "the loop ends".
3(b) 0478 7.8	Both comparisons correct, including the boundaries: `Seats >= 1` and `Seats <= 8` joined with `AND` in an `UNTIL` statement **Example:** `UNTIL Seats >= 1 AND Seats <= 8`	**2**	Accept `Seats > 0` // `Seats < 9` (Seats is an INTEGER). Accept `UNTIL NOT (Seats < 1 OR Seats > 8)` for 2 marks. Do not accept `UNTIL Seats >= 1 AND <= 8` for the second mark (each side must be a full comparison).
4(a) 0478 7.8	Any two from: Count is never changed / incremented (inside the loop) Count stays at 0 so `Count < 10` is always TRUE The loop never stops // it is an infinite loop (so line 07 is never reached)	**2**	max 2. Do not accept "the loop is wrong" on its own.
4(b) 0478 7.8	`Count ← Count + 1` Placed inside the loop // between line 03 and line 06 (e.g. after line 05, before ENDWHILE)	**2**	Accept = for ←. Do not award the second mark if the statement is placed before line 03 or after line 06.

QUESTION	ANSWER	MARKS	GUIDANCE
5(a) 0478 7.8	Any two from: The No flow line goes back to the box that sets Count and Total to 0 // the initialisation is inside the loop Count is reset to 0 every time, so it never reaches 5 // the loop never ends // infinite loop Total is reset to 0 every time, so it would only hold the last price The total is never output	**2**	max 2.
5(b) 0478 7.8	The No flow line should go back to (just before) INPUT Price // join the No flow line to the flow line between the Count ← 0 / Total ← 0 box and INPUT Price	**1**	Accept "the loop should start at INPUT Price" // "move the box that sets Count and Total to 0 out of the loop".

Total: 21 marks

MODULE K ALGORITHM DESIGN · K10

Explaining the purpose of an algorithm: mark scheme

21 marks · Cambridge 0478 syllabus 7.3 (purpose of a given algorithm), 7.4 (standard methods of solution), 7.5(a) (validation checks), 7.7 (trace tables)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. **...** means the point depends on the one before it. Credit any other valid answer that describes the algorithm accurately; do not credit vague answers that would fit any algorithm.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
0478 7.4</td><td>One mark for each correct row:
A: Totalling
B: Counting
C: Maximum
D: Average</td><td>4</td><td>A row with more than one tick scores 0. Minimum is not used.</td></tr>
<tr><td>2(a)
0478 7.7</td><td>One mark for each correct column:
<table><tr><th>Minutes</th><th>Count</th><th>Total</th><th>Number</th><th>OUTPUT</th></tr><tr><td></td><td>0</td><td>0</td><td>0</td><td></td></tr><tr><td>25</td><td></td><td>25</td><td>1</td><td></td></tr><tr><td>40</td><td>1</td><td>65</td><td>2</td><td></td></tr><tr><td>30</td><td>2</td><td>95</td><td>3</td><td></td></tr><tr><td>15</td><td></td><td>110</td><td>4</td><td></td></tr><tr><td>50</td><td>3</td><td>160</td><td>5</td><td></td></tr><tr><td></td><td></td><td></td><td></td><td>Average: 32</td></tr><tr><td></td><td></td><td></td><td></td><td>Long sessions: 3</td></tr></table></td><td>5</td><td>Minutes: 25, 40, 30, 15, 50. Count: 1, 2, 3 (30 is counted). Total: 25, 65, 95, 110, 160. Number: 1, 2, 3, 4, 5. OUTPUT: both lines, in this order, after the last input. Values may be repeated on later rows and rows may be laid out differently, as long as each column’s values are in the correct order. Ignore quotation marks. Accept 32.0.</td></tr>
<tr><td>2(b)
0478 7.3</td><td>Any two from:
Inputs the number of minutes for (each of) five sessions
Calculates and outputs the average / mean number of minutes (per session)
Counts and outputs how many sessions were 30 minutes or longer / at least 30 minutes
(Calculates) the total number of minutes</td><td>2</td><td>max 2. Do not accept “more than 30 minutes” or “over 30 minutes” (30 is counted). Do not accept “outputs 32 and 3” alone (results for one set of data, not the purpose).</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) 0478 7.3	Any four from: Asks for / inputs a part code (into `Wanted`) Repeats the input until the code is (exactly) 6 characters long Searches the `PartCode` array starting at the first element / index 1 Compares each code in turn / in order with the code input Stops when the code is found // when all 500 codes have been checked If found, outputs ("Stored in bay" and) the bay number from `Bay` at the same index If not found, outputs "Part not stocked"	**4**	max 4. Credit descriptions in words or using the identifiers. Do not accept "searches for the part" alone for the search points; the answer must show the codes are checked one at a time.
3(b) 0478 7.4	Linear search	**1**	Do not accept "search" alone or binary search.
3(c) 0478 7.5(a)	Length check	**1**	Do not accept range, presence, type or format check.
4(a) 0478 7.7	13	**1**	2 + 7 + 0 + 4 = 13.
4(b) 0478 7.3	Adds up / totals the digits of the number input and outputs the total // outputs the sum of the digits of the number	**1**	Must refer to the digits. Do not accept "adds up numbers" or "outputs Y".
4(c) 0478 7.3	Output: 0 ... the condition `X > 0` is false at the first check // the loop is pre-condition so its body is never run // so `Y` keeps its starting value (of 0)	**2**	The explanation mark depends on the output 0. Accept "the WHILE loop is skipped".

Total: 21 marks