OCR GCSE (J277)

Module M: Programming

Workbooklet mark scheme · 22 parts · 469 marks

Contents

Part	Topic	Marks
M1	Variables, constants, input and output: mark scheme	22
M3	Data types and casting: mark scheme	21
M5	Arithmetic operators, including MOD, DIV and exponentiation: mark scheme	20
M6	Comparison operators: mark scheme	22
M7	Boolean operators AND, OR and NOT in code: mark scheme	22
M8	Selection: IF, ELSE IF and ELSE: mark scheme	20
M9	Selection: switch/case statements: mark scheme	22
M10	Count-controlled loops (FOR): mark scheme	22
M11	Condition-controlled loops (WHILE): mark scheme	21
M12	Post-condition loops, and choosing the right loop: mark scheme	22
M14	Nested selection and nested loops: mark scheme	22
M16	String handling: length, substring, case, concatenation and slicing: mark scheme	22
M18	One-dimensional arrays: mark scheme	22
M19	Two-dimensional arrays and looping through them: mark scheme	21
M21	File handling: open, read, write and close: mark scheme	22
M22	Procedures: mark scheme	22
M23	Functions and return values: mark scheme	22
M24	Parameters and arguments: mark scheme	20
M25	Local and global variables: mark scheme	22
M26	Random numbers: mark scheme	20
M27	Records: mark scheme	20
M28	Writing maintainable code: mark scheme	20
	Total	**469**

MODULE M PROGRAMMING · M1

Variables, constants, input and output: mark scheme

22 marks · OCR J277 spec 2.2.1 (the use of variables, constants, operators, inputs, outputs and assignments; sequence), with 2.1.2 (trace tables, identifying errors)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative words; **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it depends on the bullet before it. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Penalise a repeated syntax slip once, then follow through.

QUESTION	ANSWER	MARKS	GUIDANCE
1 J277 2.2.1	Its value can be changed while the program is running: Variable Its value stays the same while the program is running: Constant It has an identifier (a name) that the program uses to refer to its value: Both	**3**	1 mark per correct row. A row with more than one tick scores 0 for that row.
2(a) J277 2.2.1	`RATE`	**1**	Accept `const RATE = 4` // line 01. Do not accept "4" alone (that is the value, not the constant).
2(b) J277 2.2.1	`name` // `hours` // `cost`	**1**	Any one. If more than one is given, all must be variables. Do not accept `RATE`, `input`, `print` or `int`.
2(c) J277 2.2.1	12	**1**	Accept 12.0. Do not accept `cost` or `3 * 4`.
2(d) J277 2.2.1	If the hourly rate changes, it only needs to be changed in one place / on line 01 // the rate is easier to update The name `RATE` makes it clear what the number means // the code is easier to read / understand The value cannot be changed (by mistake) while the program is running Both lines are certain to use the same value // avoids typing a different number on one of the lines	**2**	Max 2. Credit any other valid reason. Do not accept "it is faster", "it uses less memory" or "it is more secure". "Easier to change" is accepted only as updating the value in the code, not as changing it while the program runs.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(a)
J277 2.2.1
J277 2.1.2</td><td>
<table>
<tr><th>LINE</th><th>X</th><th>Y</th><th>OUTPUT</th></tr>
<tr><td>01</td><td>6</td><td></td><td></td></tr>
<tr><td>02</td><td></td><td>4</td><td></td></tr>
<tr><td>03</td><td>10</td><td></td><td></td></tr>
<tr><td>04</td><td></td><td>20</td><td></td></tr>
<tr><td>05</td><td>3</td><td></td><td></td></tr>
<tr><td>06</td><td></td><td></td><td>3</td></tr>
<tr><td>07</td><td></td><td></td><td>20</td></tr>
</table>
x: 6, 10, 3 (in this order)
y: 4, 20 (in this order)
Output: 3, 20 (in this order)
</td><td>3</td><td>1 mark per column. Line numbers are not marked. Ignore values repeated on later rows. Do not award the Output mark if extra values are output or if outputs are written as <code>print(x)</code> / x.</td></tr>
<tr><td>3(b)
J277 2.2.1</td><td>The lines run in sequence // one after another, in the order they are written // line 05 runs after line 03 (and before line 06)
Line 05 assigns 3 to x, which overwrites / replaces the 10 // a variable holds only one value at a time</td><td>2</td><td>Do not accept “x was changed” alone for the second mark: the answer must show that the old value is replaced. Do not accept “10 is stored as well” or “x is a constant”.</td></tr>
<tr><td>4
J277 2.2.1
J277 2.1.2</td><td>Line 03
… <code>pitchesLeft = PITCHES - booked</code>
Line 04
… <code>print("Pitches left")</code></td><td>4</td><td>Each correction mark depends on the matching line number. Line 03: the error is that a constant cannot be changed / assigned a new value. Allow <code>pitchesLeft = 25 - booked</code>. Do not credit changing line 01 to make <code>PITCHES</code> a variable: <code>pitchesLeft</code> would still never be assigned. Line 04: the text must be in quotes; allow single quotes; capitals ignored. Do not credit line 05 as an error.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE

5
J277 2.2.1
	Constant declared with a sensible name and the value 0.15 (e.g. `const PRICE = 0.15`)
	Number of pages input ...
	... and stored in a variable
	Total cost calculated as the number of pages multiplied by the constant (by name)
	Total cost output with a message (e.g. `"Total cost: £"`)
	Example answer (ERL):
	`const PRICE = 0.15`
	`pages = int(input("Enter the number of pages"))`
	`total = pages * PRICE`
	`print("Total cost: £" + str(total))`
		5	1 mark per bullet. Do not penalise a missing `int()` / `str()` cast (casting is 2.2.2). Allow the prompt in `input(...)` or in a `print` before it; a prompt is not required for the input marks. Calculation mark: do not award if `0.15` is typed in the calculation instead of the constant. The calculation may be done inside the output statement. Output mark: the message and the total must both be output; allow + or a comma between them. Do not accept `input` used as a variable name, or maths symbols such as × for multiply. In Python, accept a constant written as a normal variable in capitals (e.g. `PRICE = 0.15`).

Total: 22 marks

MODULE M PROGRAMMING · M3

Data types and casting: mark scheme

21 marks · OCR J277 spec 2.2.2 (the use of data types: integer, real, Boolean, character and string, casting; choosing suitable data types for data in a given scenario; data types may be temporarily changed through casting, and where this may be useful), with 2.2.1 (input, output, assignment) for Q2 and Q5

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative words; **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it depends on the bullet before it. In code answers, ignore capitals and minor syntax slips; repeated slips are penalised once, then follow through. Credit any other valid answer.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.2	Number of whole hours: integer Price per hour: real Helmet included: Boolean Frame size code: character / char	**4**	1 mark per correct row. Accept: int; float for real; bool for Boolean. Frame size: also allow string (a single letter can be stored as a string). Do not accept: number / numeric / decimal / text / letter / yes-no on their own; currency for real.
1(b) J277 2.2.2	It starts with a 0 / leading zero, which would be lost / removed if stored as an integer It contains a space (which an integer cannot hold) No calculations / arithmetic are done on a phone number It may contain other characters, e.g. + or brackets (in other phone numbers)	**2**	Max 2. Any two points. Do not accept: "it is not a number" alone; "it is easier / more efficient" alone; "a string can store letters" without linking to this phone number. Credit any other valid answer.
2(a) J277 2.2.2 J277 2.2.1	String	**1**	Accept: str. Do not accept text / characters. Do not accept integer or real: `input()` returns a string until it is cast.
2(b) J277 2.2.2	Changing / converting a value / variable from one data type to another (temporarily)	**1**	Do not accept an example on its own (e.g. 'changing a string to an integer'): it must say one data type to another. Do not accept "changing a variable" or "changing the value" alone (no mention of data type).

QUESTION	ANSWER	MARKS	GUIDANCE
2(c) J277 2.2.2	`cost = int(minutes) * 0.2`	**1**	Accept `float(minutes)` or `real(minutes)` in place of `int(minutes)`. Accept the cast applied in a separate line before the calculation, as long as line 02's calculation then works. Do not accept `str(minutes)` or casting the result instead, e.g. `int(minutes * 0.2)`.
2(d) J277 2.2.2	`print("Cost in pounds: " + str(cost))`	**1**	The cost must be cast to a string with `str()`. The message wording may differ. Do not accept `int(cost)` or `float(cost)` (still a number joined to a string).
3 J277 2.2.2	`int("15") + 5`: 20 `"15" + "5"`: "155" `str(9) + "0"`: "90" `float("1.5") * 2`: 3.0	**4**	1 mark per correct row. Row 1: 20 (not "20" and not "155"). Rows 2 and 3: allow without speech marks (155, 90). Do not accept 20 for row 2 or 9 for row 3. Row 4: accept 3 or 3.00.
4 J277 2.2.2 J277 2.2.1	`total / 4`: real (the value is 2.5) `total MOD 4`: integer (the value is 2) `total >= 10`: Boolean (the value is True)	**3**	1 mark per correct row. Accept float for real, int for integer, bool for Boolean. Do not accept the value instead of the data type (2.5, 2, True score 0).

5
J277 2.2.2
J277 2.2.1
	Input the weight and store it in a variable
	Cast / convert the weight to real using `float()` / `real()` or the language's equivalent (e.g. `Double.parseDouble`, `CDbl`, `Convert.ToDouble`)
	Calculate weight × 2.40 using the cast value
	Output the cost with a suitable message ... with the cost cast to a string if + is used in ERL / Python (allow no cast in languages that convert automatically, e.g. Java, C#)
	Example answer (ERL):
	`weight = float(input("Enter parcel weight in kg"))`
	`cost = weight * 2.40`
	`print("Postage cost: " + str(cost))`
		4	1 mark per bullet, max 4.
			Casting may be on the input line or on a separate line.
			Do not award the cast mark for `int()` (it would lose the decimal part, e.g. 3.5) or for casting the cost instead of the weight.
			Calculation: accept 2.4; accept the calculation done inside the output statement.
			Do not accept × or x as the multiplication operator in code.
			Output: a comma between the message and the cost is allowed (e.g. Python `print("Cost:", cost)`).
			Do not award the output mark if the variable name is inside the speech marks.
			A prompt for the input is not required. Any valid high-level language is accepted.

Total: 21 marks

MODULE M PROGRAMMING · M5

Arithmetic operators, including MOD, DIV and exponentiation: mark scheme

20 marks · OCR J277 spec 2.2.1 (the common arithmetic operators: + − * / MOD DIV ^), with 2.1.2 (trace tables)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. **...** means the second point depends on the first. Code answers are marked for logic, not exact syntax: any high-level language or OCR Exam Reference Language is accepted; penalise a repeated syntax slip once, then follow through.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	`17 MOD 5`: 2 `17 DIV 5`: 3 `3 ^ 4`: 81 `9 / 2`: 4.5 `2 + 3 * 4`: 14	**5**	1 mark per correct row. Correct answer only. Do not accept: 3.4 or "3 r 2" for DIV; 12 for `3 ^ 4`; 4 or "4 r 1" for `9 / 2` (accept 4½); 20 for `2 + 3 * 4` (multiplication is done before addition).

2(a) J277 2.2.1	`minutes = totalSeconds DIV 60`
	`seconds = totalSeconds MOD 60`
	`area = side ^ 2`
		3	1 mark per row. Ignore case. Accept language equivalents: `//` for DIV, `%` for MOD, `**` for ^. Do not accept `/` in row 1 (it gives a decimal, not whole minutes) or * in row 3 (`side * 2` is not the area).

3(a) J277 2.2.1	4	**1**	Items 25, 50, 75 and 100 give a remainder of 0. Accept a list of the four messages.
3(b) J277 2.2.1	Inspect item 25	**1**	Ignore quotation marks. Accept "Inspect item25" (spacing). Do not accept "Inspect item 1" or "25" alone.
3(c) J277 2.2.1	24	**1**	Items 1 to 24 give a quotient of 0 (items 25 to 100 give 1 to 4). Correct answer only.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>4(a)
J277 2.2.1, 2.1.2</td><td>boxes: 3, 6, 0 (in order)
spare: 2, 0, 5 (in order)
Output: 3 boxes, 2 spare / 6 boxes, 0 spare / 0 boxes, 5 spare (all three, in order)
<table><tr><th>DAY</th><th>EGGS</th><th>BOXES</th><th>SPARE</th><th>OUTPUT</th></tr><tr><td>1</td><td>20</td><td>3</td><td>2</td><td>3 boxes, 2 spare</td></tr><tr><td>2</td><td>36</td><td>6</td><td>0</td><td>6 boxes, 0 spare</td></tr><tr><td>3</td><td>5</td><td>0</td><td>5</td><td>0 boxes, 5 spare</td></tr></table></td><td>3</td><td>1 mark per correct column (boxes, spare, Output). The day and eggs columns are not marked. Output: ignore quotation marks and small spacing slips; FT the candidate's own boxes and spare values. Do not accept 0.0 / decimals in boxes (DIV gives a whole number), or a blank for 0 in spare on day 2 or boxes on day 3.</td></tr>
<tr><td>4(b)
J277 2.2.1</td><td>7 boxes, 5 spare</td><td>1</td><td>Ignore quotation marks. Both values needed, the right way round.</td></tr>
<tr><td>5(a)
J277 2.2.1</td><td>1 mark each to max 5:
• Asks for / inputs the number of guests and stores it (as an integer)
• Uses DIV 8 (or equivalent) to find the number of full tables
• Uses MOD 8 (or equivalent) to check whether there are guests left over ...
• ... and adds 1 table only when the remainder is not 0
• Outputs the number of tables
• Calculates the empty seats correctly and outputs them, using either method:
empty = tables × 8 − guests
OR empty = 8 − (guests MOD 8) if guests MOD 8 ≠ 0, otherwise 0
<pre>Example answer (ERL):
guests = int(input("Number of guests: "))
tables = guests DIV 8
if guests MOD 8 != 0 then
    tables = tables + 1
endif
empty = tables * 8 - guests
print("Tables needed: " + str(tables))
print("Empty seats: " + str(empty))</pre></td><td>5</td><td>Allow any logically valid method, e.g. <code>tables = (guests + 7) DIV 8</code> gains bullets 2, 3 and 4; a loop that adds a table while seats < guests gains bullets 2 to 4 if correct. Accept <code>> 0</code> for <code>!= 0</code>. Using / with no rounding up: do not award bullet 2. Adding 1 table every time (so 40 guests give 6 tables): do not award bullet 4. Empty-seat answer that gives 8 instead of 0 when guests is a multiple of 8: do not award the last bullet. Do not accept × in place of * in code. Missing <code>int()</code>/casting: do not penalise. Flowcharts or written English: 0.</td></tr>
</table>

Total: 20 marks

MODULE M PROGRAMMING · M6

Comparison operators: mark scheme

22 marks · OCR J277 spec 2.2.1 (the use of operators; recognise and use the comparison operators ==, !=, <, <=, >, >=; selection and condition-controlled iteration)

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. In code answers, accept any logically valid method and ignore minor syntax slips, but the comparison operator itself must be correct: do not accept maths symbols (≥, ≤, ≠), => or =<, or the comparison written in English. A comparison with the operands the other way round and the operator reversed is equivalent (e.g. `630 < load` = `load > 630`) and is accepted.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	1 mark per row: `!=` … Not equal to / is not the same as Greater than or equal to … `>=` `==` … Equal to / is the same as / checks whether two values are equal	**3**	Row 2: do not accept => or ≥. Row 3: do not accept "assigns / stores a value" (that is =). Do not accept "not" alone for row 1.
1(b) J277 2.2.1	Boolean	**1**	Accept Bool. Accept "Boolean (True or False)". Do not accept "True / False" alone (these are the values, not the data type).
2(a) J277 2.2.1	1 mark per row: `stock > reorder` … False `stock >= reorder` … True `size != "M"` … False `boxes * 4 <= stock` … True	**4**	Accept T / F, TRUE / FALSE, 1 / 0. Row 1 and row 2: 12 compared with 12, so only >= is True. Row 4: 3 * 4 = 12, and 12 <= 12 is True.
3(a) J277 2.2.1	`load > 630`	**1**	Accept `if load > 630 then`. Accept `load >= 631` / `load > 630.0`. Do not accept `load >= 630` (630 kg would sound the alarm).
3(b) J277 2.2.1	`weight <= 2`	**1**	Accept `weight <= 2.0`. Do not accept `weight < 2` (a 2 kg parcel would be left out) or `weight =< 2`.
3(c) J277 2.2.1	`code == "B4X9"`	**1**	Accept single quotes. The value must be in quotes (code is a string). Do not accept a single = (in OCR Exam Reference Language this is assignment).

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) J277 2.2.1 (2.1.2 logic errors)	1 mark for each line number, 1 mark for each correct corrected line: Line 02 … `if temp >= 30 then` Line 08 … `while moisture < 40`	**4**	The correction mark is for the correction of the line identified. Accept equivalents: `temp > 29` / `30 <= temp` (temp is an integer); `moisture <= 39` / `40 > moisture`. Accept the condition alone without `if`/`then`/`while`. Do not accept `temp => 30`. A correction that changes other lines (e.g. swapping the two prints) scores 0 for that error.
5(a) J277 2.2.1	1 mark per row: 70 … OK 30 … OK 29 … Too slow	**3**	Ignore quotation marks and capital letters. 70 is not more than 70, and 30 is not less than 30, so both give OK.

6(a)
J277 2.2.1
	1 mark each:
	Input of the water level, stored in a variable
	Checks if level is more than 180 (`level > 180`) and outputs Overflow warning
	Checks if level is 20 or less (`level <= 20` / `level < 21`) and outputs Refill needed
	Outputs Level normal in all other cases … with only one message output for any level (e.g. `elseif` / `else`, nested `if`s, or separate `if`s with full conditions)
	Example answer (ERL):
	`level = int(input("Enter water level"))`
	`if level > 180 then`
	`  print("Overflow warning")`
	`elseif level <= 20 then`
	`  print("Refill needed")`
	`else`
	`  print("Level normal")`
	`endif`
		4	Allow any logically valid method and any high-level language. Allow missing casting. Allow `level >= 181` for > 180. A middle check written as `level > 20 AND level <= 180` is accepted; `level > 20 AND <= 180` (no variable on the second side) is not. Do not award bullet 4 if two messages can be output for one level (e.g. 200 gives Overflow warning and Level normal). Wrong boundary (`>= 180` or `< 20`) loses that bullet. Do not accept ≥, ≤ or comparisons in English. Minor spelling slips in the messages are ignored.

Total: 22 marks

MODULE M PROGRAMMING · M7

Boolean operators AND, OR and NOT in code: mark scheme

22 marks · OCR J277 spec 2.2.1 (the common Boolean operators AND, OR and NOT; comparison operators; selection)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Capitals in keywords are ignored (`and` = `AND`). Do not accept maths symbols (≥, ≤, ≠) or English ("is more than") in place of operators.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	True only when both conditions are True: AND Reverses a Boolean value: NOT True when at least one condition is True: OR	**3**	1 mark per correct row. More than one tick in a row: 0 for that row.
2(a) J277 2.2.1	`temperature > 20 AND doorOpen`: False `temperature > 30 OR NOT doorOpen`: True `NOT(temperature < 25 AND alarmSet)`: False	**3**	1 mark per correct row. Accept F/T, 0/1. Working: row 1 True AND False = False; row 2 False OR (NOT False = True) = True; row 3 NOT(True AND True) = NOT True = False.
3(a) J277 2.2.1	Two correct comparisons: `windSpeed >= 0` and `windSpeed <= 250` … … joined with `AND` Example: `windSpeed >= 0 AND windSpeed <= 250`	**2**	Both boundaries must be included: `> 0` or `< 250` does not get the first mark. As readings are whole numbers, accept `windSpeed > -1` and `windSpeed < 251`. Each side must be a full comparison: `windSpeed >= 0 AND <= 250` = 1 (AND mark only). Accept `NOT(windSpeed < 0 OR windSpeed > 250)` for 2. Do not accept OR joining the two in-range checks (always True): max 1.
3(b) J277 2.2.1	Two correct comparisons: `windSpeed < 0` and `windSpeed > 250` … … joined with `OR` Example: `windSpeed < 0 OR windSpeed > 250`	**2**	Accept `windSpeed <= -1` and `windSpeed >= 251` (whole numbers). `<= 0` or `>= 250` (rejects a valid boundary) does not get the first mark. AND in place of OR (never True): max 1 for the comparisons. The question asks for OR: a correct `NOT(... AND ...)` with no OR gets max 1.

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) J277 2.2.1	Line 03 (identified) ... ... `if hour >= 7 AND hour <= 19 then` (OR changed to AND) Line 04 (identified) ... ... `if keycard == "Y" OR keycard == "y" then` (second comparison given in full)	**4**	The correction mark for each error needs the right line number. Accept `hour > 6 AND hour < 20`. Line 04: accept `keycard.upper == "Y"` or equivalent; accept a line that adds the missing `keycard ==` even if `then` is omitted. Do not accept changing OR to AND on line 04 (a keycard cannot be both "Y" and "y"). Do not accept changes to lines 01, 02 or the outputs.
4(b) J277 2.2.1	`hour <= 19` is True (when hour is 2) / one of the two comparisons is True OR gives True when at least one condition is True / OR only needs one side to be True ... ... so the whole condition on line 03 is True, and the `else` on line 09 is never reached	**2**	Max 2. Accept "every hour is either 7 or more, or 19 or less, so the condition is always True" for 2. "It should be AND" alone (no reason) = 0. Do not credit answers about line 04.

5(a)
J277 2.2.1
	1 mark each to max 6:
	Inputs the battery level and the booked answer, storing each in a variable
	Attempt at selection (`if ... then ... endif` or equivalent)
	Correct battery check: `battery >= 20` / `battery > 19`
	Correct not-booked check: `booked == "N"` / `booked != "Y"` / `NOT(booked == "Y")`
	Both checks joined with AND (or correctly nested IFs) so "Available" only when both are met
	Outputs `Available` and `Not available` in the correct cases (including every failing case)
	Example:
	`battery = int(input("Enter battery level"))`
	`booked = input("Is the bike booked? (Y/N)")`
	`if battery >= 20 AND booked == "N" then`
	`    print("Available")`
	`else`
	`    print("Not available")`
	`endif`
		6	Do not penalise a missing `int()`/casting. Accept any variable names except reserved words (e.g. `input`). `battery > 20` loses the battery-check mark. Second side with no variable (`battery >= 20 AND == "N"`) loses the not-booked mark. Using OR between the two checks loses the AND mark. Nested IFs: the output mark needs "Not available" in both the low-battery and the booked case. Accept `booked == "N"` or `booked == "n"`. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M8

Selection: IF, ELSE IF and ELSE: mark scheme

20 marks · OCR J277 spec 2.2.1 (the three programming constructs: selection; comparison operators; inputs and outputs), 2.1.2 (nesting for selection; identify logic errors and suggest fixes; complete, write and refine algorithms)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers: mark the logic, not the language. Allow OCR Exam Reference Language or any high-level language, and allow any logically valid method. Do not penalise minor syntax slips (missing casting, capitals, a missing colon, `else if` or `elif` for `elseif`). A recurring slip: penalise once, then follow through. Do not accept maths symbols such as ≥ or ≤ in code. Outputs: ignore quotation marks round output text.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	1 mark each to max 2: A condition is tested / checked / evaluated (as true or false) The result decides which block of code / which statements run (next) / which path the program takes Code may be skipped / only some of the code runs	**2**	An example alone, e.g. "an if statement" or "if ... else": max 1. Do not accept answers that describe repeating code (iteration) or running every line in order (sequence). Credit any other valid point.
2(a) J277 2.2.1, 2.1.2	1 mark per correct output: 22: `Fan low` 30: `Fan high` 40: `Fan high` 15: `Fan off`	**4**	Spelling of the output must be clear. Do not accept more than one output in a row. Do not accept `print("Fan low")`.
2(b) J277 2.2.1	The condition on line 02 is tested first / is true first, and once one branch runs the rest of the statement (the `elseif`/`else` parts) is skipped / only the first true branch runs / line 04 is never tested	**1**	Do not accept "because 45 is more than 40" alone (does not explain why line 04 does not also run).
3(a) J277 2.1.2, 2.2.1	`Bronze`	**1**	Do not accept `Silver` (that is the intended badge, not the output).
3(b) J277 2.1.2, 2.2.1	Any number that is 8 or more (or 12 or more) is also 4 or more, so the condition on line 02 is true first / is tested first and the `elseif` lines are skipped / lines 04 and 06 are never reached when they could be true	**1**	Allow "the tests are in the wrong order; `>= 4` is checked first, so a member with 8+ classes gets Bronze". Do not accept "it is a logic error" or "the order is wrong" alone.

QUESTION	ANSWER	MARKS	GUIDANCE

3(c)
J277 2.1.2, 2.2.1
	1 mark each to max 2:
	`classes >= 12` tested first and gives `Gold`
	... then `classes >= 8` gives `Silver` and then `classes >= 4` gives `Bronze` (in that order, using `elseif`)
	`02 if classes >= 12 then`
	`03  badge = "Gold"`
	`04 elseif classes >= 8 then`
	`05  badge = "Silver"`
	`06 elseif classes >= 4 then`
	`07  badge = "Bronze"`
		2	Allow conditions with full ranges in any order, e.g. `elseif classes >= 8 AND classes <= 11 then` (or `< 12`); 1 mark for Gold correct, 1 mark for Silver and Bronze correct. Allow `> 11`, `> 7`, `> 3`. Each side of AND must be a full comparison: do not accept `classes >= 8 AND <= 11`. Allow a rewrite that also keeps lines 08 to 10.

4(a) J277 2.1.2, 2.2.1	1 mark per correct output: 600, "Y": `Lift moving` 720, "N": `Close the door` 650, "Y": `Overloaded`	**3**	Row 2: do not accept `Overloaded` or two outputs (the inner `if` is never reached when the door is open).

5(a)
J277 2.2.1, 2.1.2
	1 mark each to max 6:
	Inputs the length and stores it in a variable
	Attempt at selection with more than two branches (`if` ... `elseif` / nested `if`)
	Outputs `Invalid` for a length of 0 or less (`<= 0`)
	Outputs `Small` for more than 0 and up to 30 (`<= 30`) and `Medium` for more than 30 and up to 50 (`<= 50`)
	Outputs `Large` for more than 50 and up to 80 (`<= 80`)
	Outputs `Too big` for more than 80 (`else` / `> 80`)
	Only one message can be output for any length (conditions in a working order / ranges that do not overlap)
	`length = float(input("Longest side in cm"))`
	`if length <= 0 then`
	`  print("Invalid")`
	`elseif length <= 30 then`
	`  print("Small")`
	`elseif length <= 50 then`
	`  print("Medium")`
	`elseif length <= 80 then`
	`  print("Large")`
	`else`
	`  print("Too big")`
	`endif`
		6	Boundaries must be exact: 30 is Small, 50 is Medium, 80 is Large, 0 is Invalid. Allow `int(...)` or no casting. Allow separate `if` statements with full ranges using AND (e.g. `if length > 30 AND length <= 50 then`). Separate `if` statements with open conditions (e.g. `if length <= 50` after `if length <= 30`) give two outputs: do not award the last bullet. Allow tests in reverse order (e.g. `> 80` first). A message with the output is not needed.

Total: 20 marks

MODULE M PROGRAMMING · M9

Selection: switch/case statements: mark scheme

22 marks · OCR J277 spec 2.2.1 (the three basic programming constructs: selection; ERL CASE SELECT or SWITCH), with 2.1.2 (interpret, correct, complete and refine algorithms; identify errors)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language (e.g. Python `match/case` with `case _:` as the default, Visual Basic `Select Case` ... `Case Else` ... `End Select`, C#/Java `switch` with `break`). Minor syntax slips (a missing colon, `break` present or absent) are not penalised; a repeated slip is penalised once then follow through. Capitals in keywords are ignored. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	Paper tray from a size code: switch Parcel over 20 kg: if Fan speed from a temperature range: if	**3**	1 mark per correct row. More than one tick in a row: 0 for that row. Row 1 compares one variable with a few fixed values. Rows 2 and 3 need a comparison (>, <) or a range of real values, which a case with one fixed value cannot test.
2(a) J277 2.2.1, 2.1.2	`P`: Blue bin `M`: Grey bin `g`: Ask staff	**3**	1 mark per correct row. Ignore quotation marks round the output. Do not accept `print("Blue bin")` written as the output. Do not accept "Green bin" for `g`.
2(b) J277 2.2.1	Max 2: `"g"` does not match `"G"` / the comparison is case sensitive / lower case and upper case letters are different values no case matches (none of `"G"`, `"P"`, `"M"`) ... ... so the `default` branch (line 09/10) runs	**2**	"It is not one of the options" with "so default runs" = 2. "It is lower case" alone = 1. "The program has an error" = 0 (the program works as written).

QUESTION	ANSWER	MARKS	GUIDANCE

3(a)
J277 2.2.1, 2.1.2
	1 mark each to max 4:
	`switch exhibit:` … `endswitch` (switch on the variable `exhibit`, with a matching end)
	`case 1:` followed by `print("Fossils")`
	`case 2:` with `print("Ancient coins")` **and** `case 3:` with `print("Clocks")`
	`default:` with `print("No guide for this number")`, as the last option
	Example:
	`switch exhibit:`
	`  case 1:`
	`  print("Fossils")`
	`  case 2:`
	`  print("Ancient coins")`
	`  case 3:`
	`  print("Clocks")`
	`  default:`
	`  print("No guide for this number")`
	`endswitch`
		4	Case values in quotes (`"1"`) would not match a whole number: penalise once then FT. Do not accept the answer written again as `if/elseif`: 0. Do not accept `case exhibit == 1:` for the case marks (penalise once then FT). A `default` that is not the last option loses the last mark. Python: `match exhibit:` … `case _:` is fine.

4(a) J277 2.1.2, 2.2.1	Line 07 (identified) … … `case 3:`	**2**	The correction mark needs line 07. Line 07 has a second `case 2:`, so the zone 3 fare (5.00) can never be set. Do not accept changes to line 08 (5.00 is the correct zone 3 fare) or line 05.
4(b) J277 2.1.2, 2.2.1	`Invalid zone` `Fare: 0`	**2**	1 mark per output line. Ignore quotation marks. Allow `Fare: 0.0` / `Fare: 0.00`. Allow both outputs on one line. Do not accept `Fare: 5.0` (that is the output after the fix).

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
J277 2.2.1
	1 mark each to max 6:
	Inputs the mode letter and stores it in a variable
	`switch` on that variable with a matching `endswitch` (or equivalent: `match`, `Select Case`)
	`case "E":` outputs `Eco: 18 degrees`
	`case "C":` outputs `Comfort: 21 degrees`
	`case "B":` outputs `Boost: 24 degrees`
	`default:` outputs `Invalid mode`
	Example:
	`mode = input("Enter mode letter")`
	`switch mode:`
	`  case "E":`
	`  print("Eco: 18 degrees")`
	`  case "C":`
	`  print("Comfort: 21 degrees")`
	`  case "B":`
	`  print("Boost: 24 degrees")`
	`  default:`
	`  print("Invalid mode")`
	`endswitch`
		6	Case values must be in quotes (text); missing quotes: penalise once then FT. Ignore small differences in the output text (e.g. a missing colon). Allow extra cases for lower-case letters, or converting the input to upper case first. A correct program written with `if/elseif/else` and no switch: max 4 (input mark and the three letter outputs). Do not accept `input` as a variable name. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M10

Count-controlled loops (FOR): mark scheme

22 marks · OCR J277 spec 2.2.1 (the use of the three basic programming constructs: iteration, count- and condition-controlled loops; variables, operators, inputs, outputs and assignments), with 2.1.2 (trace tables, identifying common errors) for Q3 and Q5

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative words; **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it depends on the bullet before it. In code answers, ignore capitals and minor syntax slips; repeated slips are penalised once, then follow through. Allow any logically valid method. Credit any other valid answer.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	A loop that repeats (a block of code) a set / fixed / known / given number of times	**1**	Accept: "a loop where the number of repetitions is known before the loop starts"; "a for loop that runs from a start value to an end value". Do not accept: "a loop that repeats" / "a loop that counts" alone; "repeats until a condition is met" (condition-controlled).
1(b) J277 2.2.1	Print 5 copies: count-controlled Ask for the password until correct: condition-controlled	**2**	1 mark per correct row. A row with both boxes ticked scores 0.
2 J277 2.2.1	`for k = 0 to 6`: 7 `for k = 5 to 20 step 5`: 4 `for k = 9 to 1 step -2`: 5	**3**	1 mark per correct row. Row 1: k takes 0, 1, 2, 3, 4, 5, 6 (both ends included). Do not accept 6. Row 2: k takes 5, 10, 15, 20. Do not accept 3 or 15. Row 3: k takes 9, 7, 5, 3, 1. Do not accept 4 or 8. Accept the answer written in words. Do not accept a list of the values instead of the number of times.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr>
<td>3
J277 2.2.1
J277 2.1.2</td>
<td>week: 1, 2, 3, 4
sold: 3, 6, 9, 12
stock: 20, 17, 11, 2 (and no further change)
Output: Left: 2 (once, at the end)
Example completed table:
<table>
<tr><th>LINE</th><th>WEEK</th><th>SOLD</th><th>STOCK</th><th>OUTPUT</th></tr>
<tr><td>01</td><td></td><td></td><td>20</td><td></td></tr>
<tr><td>02</td><td>1</td><td></td><td></td><td></td></tr>
<tr><td>03</td><td></td><td>3</td><td></td><td></td></tr>
<tr><td>05</td><td></td><td></td><td>17</td><td></td></tr>
<tr><td>02</td><td>2</td><td></td><td></td><td></td></tr>
<tr><td>03</td><td></td><td>6</td><td></td><td></td></tr>
<tr><td>05</td><td></td><td></td><td>11</td><td></td></tr>
<tr><td>02</td><td>3</td><td></td><td></td><td></td></tr>
<tr><td>03</td><td></td><td>9</td><td></td><td></td></tr>
<tr><td>05</td><td></td><td></td><td>2</td><td></td></tr>
<tr><td>02</td><td>4</td><td></td><td></td><td></td></tr>
<tr><td>03</td><td></td><td>12</td><td></td><td></td></tr>
<tr><td>08</td><td></td><td></td><td></td><td>Left: 2</td></tr>
</table>
</td>
<td>4</td>
<td>1 mark for each correct set of changes to a variable, in the correct order, and 1 mark for the output.
Line numbers are not marked. Values may be repeated on later rows. Several changes may be written on one row.
In week 4, 12 is not less than 2, so line 05 does not run and stock stays 2. Do not award the stock mark if a further value (e.g. −10) is shown.
Ignore a final value of 5 for week.
Output: accept Left: 2 with or without speech marks; do not accept print(...) written around it or 2 alone.</td>
</tr>
<tr>
<td>4
J277 2.2.1</td>
<td>A for loop with start value 60 and end value 0 ...
... and step −10
Outputs the loop variable inside the loop and outputs Lights off once, after the loop
Example answer (ERL):
<pre>for count = 60 to 0 step -10
    print(count)
next count
print("Lights off")</pre>
</td>
<td>3</td>
<td>1 mark per bullet, max 3.
Allow any loop that gives the same output, e.g. for count = 6 to 0 step -1 with print(count * 10) (award bullets 1 and 2 for a correct loop of this kind), or Python for count in range(60, -1, -10):.
Do not award bullet 3 if Lights off is output inside the loop (it would be output 7 times).
Any form is accepted for this algorithm: ERL, a high-level language or a flowchart, as long as a count-controlled loop is used.</td>
</tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
5 J277 2.1.2 J277 2.2.1	Line 02 … `for pots = 1 to 6` Line 03 … `cost = price * pots`	**4**	1 mark for each line number and 1 mark for each matching correction. A correction is only credited with its correct line number. Line 02: the loop must run for `pots` = 1 to 6 inclusive. Do not accept `0 to 5` or `1 to 7` (these output the wrong numbers of pots). Line 03: accept `cost = pots * price`. Do not credit line 04 or 05 (they are correct as written).

6
J277 2.2.1
	Input the number of parcels and store it in a variable
	A `for` loop that runs that number of times (e.g. `1 to number` // `0 to number - 1`)
	Input a weight inside the loop
	Add each weight to a running total inside the loop …
	… output the total after the loop has finished
	Example answer (ERL):
	`parcels = int(input("How many parcels have arrived?"))`
	`total = 0`
	`for count = 1 to parcels`
	`  weight = float(input("Enter weight in kg"))`
	`  total = total + weight`
	`next count`
	`print("Total weight: " + str(total) + " kg")`
		5	1 mark per bullet, max 5.
			Loop: the end value must use the number entered, not a fixed number. Python `for count in range(parcels):` is accepted.
			Total: do not award bullet 4 if the total is reset to 0 inside the loop. An uninitialised total may be taken as 0.
			Output: do not award bullet 5 if the total is output inside the loop (it would be output after every parcel). A comma between the message and the total is allowed.
			Missing casts are not penalised. Any valid high-level language is accepted.

Total: 22 marks

MODULE M PROGRAMMING · M11

Condition-controlled loops (WHILE): mark scheme

21 marks · OCR J277 spec 2.2.1 (the use of the three basic programming constructs: iteration, count- and condition-controlled loops; ERL `while ... endwhile`, condition checked before entering the loop), with 2.1.2 (trace tables; identify common errors and suggest fixes; complete and refine algorithms) and 2.3.1 (input validation) for Q4

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative words; **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it depends on the bullet before it. In code answers, ignore capitals and minor syntax slips; repeated slips are penalised once, then follow through. Any high-level language is accepted where code is written. Credit any other valid answer.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1
J277 2.2.1</td><td>Condition checked before the code inside runs: True
Code inside always runs at least once: False
Suitable when the number of repeats is not known in advance: True</td><td>3</td><td>1 mark per correct row. A row with two ticks scores 0. Row 2 is false: a <code>while</code> loop checks its condition first, so if the condition is False at the start the code inside runs zero times.</td></tr>
<tr><td>2(a)
J277 2.1.2
J277 2.2.1</td><td><code>total</code>: 0, 30, 75, 95
<code>boxes</code>: 0, 1, 2, 3
<code>weight</code>: 30, 45, 20, 15
Output: 3 (once, at the end only)
Example completed table:
<table>
<tr><th>LINE</th><th>TOTAL</th><th>BOXES</th><th>WEIGHT</th><th>OUTPUT</th></tr>
<tr><td>01</td><td>0</td><td></td><td></td><td></td></tr>
<tr><td>02</td><td></td><td>0</td><td></td><td></td></tr>
<tr><td>03</td><td></td><td></td><td>30</td><td></td></tr>
<tr><td>05</td><td>30</td><td></td><td></td><td></td></tr>
<tr><td>06</td><td></td><td>1</td><td></td><td></td></tr>
<tr><td>07</td><td></td><td></td><td>45</td><td></td></tr>
<tr><td>05</td><td>75</td><td></td><td></td><td></td></tr>
<tr><td>06</td><td></td><td>2</td><td></td><td></td></tr>
<tr><td>07</td><td></td><td></td><td>20</td><td></td></tr>
<tr><td>05</td><td>95</td><td></td><td></td><td></td></tr>
<tr><td>06</td><td></td><td>3</td><td></td><td></td></tr>
<tr><td>07</td><td></td><td></td><td>15</td><td></td></tr>
<tr><td>09</td><td></td><td></td><td></td><td>3</td></tr>
</table></td><td>4</td><td>1 mark for each correct set of changes to a variable, in the correct order, and 1 mark for the output.
Line numbers are not marked. Values may be repeated on later rows. Values may be on fewer rows (e.g. one row per pass of the loop) as long as each sequence is correct.
Do not award the <code>total</code> mark if 110 appears (95 + 15 fails the condition, so line 05 does not run again).
Do not award the output mark if anything other than 3 is output, or if 3 is output more than once.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
2(b) J277 2.2.1	0	**1**	Correct answer only. (0 + 120 <= 100 is False, so the loop body never runs and `boxes` is still 0.) Do not accept "nothing" / "no output" / 1.
2(c) J277 2.2.1	`weight` must have a value before the condition (on line 04) is checked // the condition uses `weight`, and the condition is checked before the loop is entered // to get the first box weight to test in the condition	**1**	Must link the input to the condition / the check at the start of the loop. Do not accept "to input a weight" alone (NE), or "to start the loop" alone. Accept "otherwise there would be an error because `weight` has no value when line 04 is reached".
3(a) J277 2.1.2 J277 2.2.1	`saved` is never changed / increased inside the loop // `saved` stays at 0 ... so the condition `saved < TARGET` is always True / never becomes False The loop never ends / is an infinite loop (so line 07 is never reached)	**2**	Max 2. Accept "0 is always less than 500" for the second mark. "It is an infinite loop" with no reason scores 1 mark only. Do not accept "`weeks` never changes" (it does change).
3(b) J277 2.1.2	`saved = saved + 60`	**1**	Accept `saved += 60` (high-level language). Do not accept `saved = 60`, `saved =+ 60`, `saved + 60` on its own, or `TARGET = TARGET - 60` (a constant cannot be changed).
3(c) J277 2.2.1	Weeks needed: 9	**1**	Accept 9 on its own. After 8 weeks `saved` is 480 (still less than 500); after 9 weeks it is 540. Do not accept 8 or 8.33.

QUESTION	ANSWER	MARKS	GUIDANCE

4
J277 2.2.1
J277 2.3.1
	Line 02: `temp < 10` // `temp <= 9`
	... `OR temp > 30` // `OR temp >= 31`
	Line 04: `temp = int(input("Enter temperature setting"))`
	Example answer:
	`02 while temp < 10 OR temp > 30`
	`04  temp = int(input("Enter temperature setting"))`
		3	Line 02: accept `NOT(temp >= 10 AND temp <= 30)` for both marks. Accept `or` in lower case.
			Do not award the second mark if AND is used (the condition could never be True), or if the variable is missing from the second half (`temp < 10 OR > 30`): max 1.
			Do not accept maths symbols such as ≤ or ≥.
			Line 04: must assign the input to `temp`; the prompt wording may differ. Casting is not required for the mark.
			Do not accept `input = ...` or an input that is not stored in `temp`.

5
J277 2.2.1
	Running total set to 0 before the loop
	First distance input (and stored) before the loop
	`while` loop with a correct condition, e.g. `distance != 0` // `distance > 0`
	... distance added to the total inside the loop
	... next distance input inside the loop
	Total output with a message after / outside the loop
	Example answer (ERL):
	`total = 0`
	`distance = int(input("Enter distance in metres (0 to finish)"))`
	`while distance != 0`
	`  total = total + distance`
	`  distance = int(input("Enter distance in metres (0 to finish)"))`
	`endwhile`
	`print("Total distance rowed: " + str(total) + " metres")`
		5	1 mark per bullet, max 5.
			Allow a loop such as `while True` with the input first inside the loop and an `if distance == 0 ... break`: award the "input before the loop" and "while condition" marks for this.
			A `do ... until` loop: credit the other features shown, but not the `while` loop mark.
			Uninitialised total: do not award mark 1. Do not award the total mark for `total = distance` or `total =+ distance`.
			Output mark: not awarded if the output is inside the loop.
			A comma between message and total is allowed (e.g. Python `print("Total:", total)`). Casting is not required for any mark.

Total: 21 marks

MODULE M PROGRAMMING · M12

Post-condition loops, and choosing the right loop: mark scheme

22 marks · OCR J277 spec 2.2.1 (iteration: count- and condition-controlled loops), 2.1.2 (trace tables; refine algorithms; OCR Exam Reference Language)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Capitals in keywords are ignored. Do not accept maths symbols (≥, ≤, ≠) or English ("is more than") in place of operators. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	Condition checked before the code inside runs: `while` Code inside always runs at least once: `do ... until` It is a condition-controlled loop: Both	**3**	1 mark per correct row. More than one tick in a row: 0 for that row.
2(a) J277 2.1.2, 2.2.1	1 mark each: `temp`: 78, 60, 42, 24 (after the given 96) `minutes`: 1, 2, 3, 4 (after the given 0) Outputs 78, 60, 42, 24 (in this order) Final output `Minutes to cool: 4` (as the last output, once only)	**4**	Each mark is for the whole sequence of values in that column, in order. Values may be repeated on later rows, and changes may be shown on separate rows. Ignore quotation marks round outputs. Do not award the output marks if `print(...)` is written round the values. Extra values after 24 (e.g. a further 6 or another pass of the loop) lose the mark for that column. Accept `Minutes to cool:4` (spacing).
2(b) J277 2.2.1	17 `Minutes to cool: 1`	**2**	1 mark each. Working: the loop body runs once before the condition is checked, so `temp` becomes 35 − 18 = 17 and `minutes` becomes 1; then 17 < 40 is True, so the loop ends. "Nothing is output" / only `Minutes to cool: 0` = 0 (treats the loop as checked first).

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.2.1	Lighthouse: `for` (count-controlled) Quiz machine: `do ... until` Robot vacuum cleaner: `while`	**3**	1 mark per row. Row 1: accept "count-controlled". Rows 2 and 3: "condition-controlled" alone = 0 (it does not say which loop). Row 2: do not accept `while` (the task must run at least once). Row 3: do not accept `do ... until`.
3(b) J277 2.2.1	1 mark each to max 2: A `do ... until` loop runs the code inside at least once / checks its condition only after the code has run ... ... so the robot would move forward (at least once) before the bumper is checked The bumper may already be pressed / the robot may already be against a wall, so it should not move at all / the loop needs to be able to run zero times A `while` loop checks the bumper first, so the robot does not move if it is already touching the wall	**2**	Max 2. The second bullet depends on the first being stated or clearly implied. "A while loop is better" alone (no reason) = 0. Accept "it would crash / push into the wall" as the consequence for the second bullet.

4(a)
J277 2.2.1, 2.1.2
	1 mark each:
	`level` initialised before the loop (to 1, or 0 if incremented before the output)
	`do ... until` structure with the output and `level = level + 1` both inside the loop
	Correct `until` condition so the loop stops after `Level 4` is output, e.g. `until level > 4` / `until level == 5` / `until level >= 5`
	Example:
	`level = 1`
	`do`
	`  print("Level " + str(level))`
	`  level = level + 1`
	`until level > 4`
	Alternative: `level = 0`, then inside the loop `level = level + 1` before the print, `until level == 4`.
		3	The output must be exactly `Level 1` to `Level 4`. `until level == 4` with the increment after the print outputs only Levels 1 to 3: loses the third mark. Do not award the second mark for a `while` or `for` loop (the question asks for `do ... until`); the first and third marks can still be given if the logic is the same. Accept `level += 1`. Ignore a missing `str()`.

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
J277 2.2.1 (2.3.1 input validation)
	1 mark each to max 5:
	Inputs the target temperature and stores it in a variable
	Attempt at a condition-controlled loop (`do ... until` or `while ... endwhile`) ...
	... with the input inside the loop, so the user is asked again each time
	Correct range test with both boundaries included: `target >= 10 AND target <= 30` (or, for invalid, `target < 10 OR target > 30`)
	Loop condition the right way round for the loop used (a `do ... until` ends when the target is valid; a `while` repeats while it is invalid) and `Target set to` with the target output once, after the loop
	Example:
	`do`
	`  target = int(input("Enter target temperature (10 to 30)"))`
	`until target >= 10 AND target <= 30`
	`print("Target set to " + str(target))`
	Example with a `while` loop:
	`target = int(input("Enter target temperature (10 to 30)"))`
	`while target < 10 OR target > 30`
	`  target = int(input("Must be 10 to 30. Enter again"))`
	`endwhile`
	`print("Target set to " + str(target))`
		5	Allow any logically valid method. A `while` answer that inputs before the loop and again inside it gets the "input inside the loop" mark. Accept `target > 9 AND target < 31`. Each side of a compound condition must be a full comparison: `target >= 10 AND <= 30` loses the range mark. OR joining the two valid checks / AND joining the two invalid checks loses the range mark. A single `if` with no loop (only one re-entry): max 2 (input mark and range mark). Accept a Python `while True:` loop with `break` when valid. Do not penalise a missing `int()`. Do not accept `input` as a variable name. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M14

Nested selection and nested loops: mark scheme

22 marks · OCR J277 spec 2.1.2 (use of nesting for selection and iteration; trace tables; identify errors and suggest fixes), 2.2.1 (selection and iteration), 2.2.3 (2D arrays)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Capitals in keywords are ignored. Do not accept maths symbols (≥, ≤, ≠) or English ("is less than") in place of operators.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
J277 2.1.2</td><td>Code A: Nested selection
Code B: Nested iteration
Code C: No nesting</td><td>3</td><td>1 mark per correct row. More than one tick in a row: 0 for that row. Code C has two loops one after the other (sequence), not one inside the other.</td></tr>
<tr><td>2(a)
J277 2.1.2, 2.2.1</td><td>1 mark for each correct column:
<code>row</code>: 1, 2
<code>seat</code>: 1, 2, 3, 1, 2, 3
<code>total</code>: 0, 1, 4, 6, 12
Output: 12 (once, at the end)
Example completed table:
<table>
<tr><th>ROW</th><th>SEAT</th><th>TOTAL</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td></td></tr>
<tr><td>1</td><td>1</td><td>1</td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td></tr>
<tr><td></td><td>3</td><td>4</td><td></td></tr>
<tr><td>2</td><td>1</td><td>6</td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td></tr>
<tr><td></td><td>3</td><td>12</td><td></td></tr>
<tr><td></td><td></td><td></td><td>12</td></tr>
</table></td><td>4</td><td>Values must be in the correct order. Values may be repeated on later rows (e.g. <code>row</code> 1 written on every row of the first pass). Allow the starting 0 for <code>total</code> to be missing if 1, 4, 6, 12 are correct. Do not penalise a final <code>seat</code> value of 4 or <code>row</code> value of 3 (loop variable after the loop). The output mark is lost if any other value appears in the Output column. Working: row 1 adds 1×1 and 1×3; row 2 adds 2×1 and 2×3; seat 2 is skipped each time.</td></tr>
<tr><td>3(a)
J277 2.1.2</td><td>12, "M": Use a standard locker
25, "L": Take to counter
20, "L": Use a large locker</td><td>3</td><td>1 mark per correct row. Ignore quotation marks. Accept clear shortened answers ("standard locker", "counter", "large locker"). Row 3: 20 <= 20 is True, so the inner <code>if</code> is tested.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) J277 2.1.2	Max 1: The condition on line 03 is False (25 is not 20 or less), so the `else` on line 09 / line 10 runs instead Line 04 is nested inside the True branch of the `if` on line 03 / the inner `if` is only checked when the outer condition is True	**1**	"Because it is over 20" alone = 0 unless it links to the line 03 condition being False / the inner if being skipped. Credit any other valid answer that explains the nesting.
4(a) J277 2.1.2, 2.2.3	Line 03 (identified) ... ... `for shelf = 0 to 5` Line 09 (identified) ... ... `next aisle`	**4**	The correction mark for each error needs the right line number. Line 03: accept `for shelf = 0 to 6 - 1` or any expression equal to 5. Line 09: the outer loop must be closed with its own variable. Do not accept changing line 07 (it correctly closes the inner loop), line 01 or line 02. Do not accept moving line 02 outside the outer loop (the count must restart for each aisle).
4(b) J277 2.1.2	24	**1**	4 aisles × 6 shelves = 24. Accept "4 × 6".

5(a)
J277 2.1.2, 2.2.1
	1 mark each to max 6:
	Outer loop for the levels, running 1 to 3 (3 times)
	Inner loop for the spaces, running 1 to 20 (20 times), inside the outer loop and closed before it
	Outputs the level, a dash and the space number joined together, inside the inner loop
	Attempt at selection inside the inner loop that checks the space number
	Correct condition `space <= 4` / `space < 5` ... adding `(badge)` only for those spaces and the plain code for all others
	Counter set to 0 before the loops, add 1 in the badge branch, and output after both loops end
	Example:
	`count = 0`
	`for level = 1 to 3`
	`  for space = 1 to 20`
	`  code = str(level) + "-" + str(space)`
	`  if space <= 4 then`
	`  print(code + " (badge)")`
	`  count = count + 1`
	`  else`
	`  print(code)`
	`  endif`
	`  next space`
	`next level`
	`print("Badge spaces: " + str(count))`
		6	Allow 0 to 2 and 0 to 19 loops if the output adds 1 so the codes shown are 1 to 3 and 1 to 20. Do not penalise missing `str()` or a comma separator in the output. A counter that is reset inside a loop, or output inside a loop, loses the last mark. A fixed total typed in (e.g. `print(12)`) with no counting does not get the last mark. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M16

String handling: length, substring, case, concatenation and slicing: mark scheme

22 marks · OCR J277 spec 2.2.3 (basic string manipulation: concatenation, slicing), with 2.2.2 (casting) and 2.1.2 (trace tables, logic errors)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Quotation marks around string values are not needed. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language (e.g. `len(town)`, `town[0:3]`, `town.upper()`), and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip is penalised once then follow through. Credit any other valid answer.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
J277 2.2.3</td><td>`stop.length`: 14
`stop.substring(2, 5)`: rbour
`stop.right(6)`: Lights
`stop.upper`: HARBOUR LIGHTS
`stop.left(1) + str(stop.length)`: H14</td><td>5</td><td>1 mark per correct row. Case must be exact (e.g. <code>Rbour</code>, <code>lights</code> = 0). Row 1: the space is a character, so 13 = 0. Row 2: <code>arbou</code> (counting from 1) = 0. Row 4: the space between the words must be kept. Row 5: <code>H 14</code> or <code>H + 14</code> = 0; 15 (added as numbers) = 0.</td></tr>
<tr><td>2(a)
J277 2.2.3 (ERL ASC)</td><td>68</td><td>1</td><td>Do not accept <code>"D"</code> or the binary pattern alone.</td></tr>
<tr><td>2(b)
J277 2.2.3 (ERL CHR)</td><td>e (lower case)</td><td>1</td><td>Accept <code>"e"</code> / <code>'e'</code>. Upper case <code>E</code> = 0.</td></tr>
<tr><td>3(a)
J277 2.2.3, 2.1.2</td><td>1 mark for each correct column:
<code>i</code>: 0, 1, 2, 3, 4
<code>letter</code>: K, 4, (space), 7, T
<code>clean</code>: (empty), K, K4, K47, K47T
Output: k47t (once, at the end)
<table><tr><th>I</th><th>LETTER</th><th>CLEAN</th><th>OUTPUT</th></tr><tr><td></td><td></td><td>(empty)</td><td></td></tr><tr><td>0</td><td>K</td><td>K</td><td></td></tr><tr><td>1</td><td>4</td><td>K4</td><td></td></tr><tr><td>2</td><td>(space)</td><td>K4</td><td></td></tr><tr><td>3</td><td>7</td><td>K47</td><td></td></tr><tr><td>4</td><td>T</td><td>K47T</td><td></td></tr><tr><td></td><td></td><td></td><td>k47t</td></tr></table></td><td>4</td><td>Values must be in the correct order. The initial empty value of <code>clean</code> may be left blank or shown as <code>""</code>. Repeating <code>K4</code> on the space row, or leaving that cell blank, are both fine. For <code>letter</code> at i = 2 accept a blank-looking cell, <code>" "</code>, “space” or a symbol for a space. Output <code>K47T</code> (not lower case) = 0 for the Output mark; quotation marks around the output are ignored. A space kept in <code>clean</code> (e.g. <code>K4 7T</code>) loses the <code>clean</code> mark. A value of i = 5 loses the <code>i</code> mark.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) J277 2.2.3, 2.1.2	Line 02 (identified) ... ... `first = code.substring(0, 1)` / `first = code.left(1)` Line 03 (identified) ... ... `if code.length == 6 AND first == "X" then`	**4**	The correction mark for each error needs the right line number. Line 02: accept `code[0]` / `code[0:1]`. Line 03: accept `code.length >= 6 AND code.length <= 6`; do not accept `>= 6` alone (accepts 7 or more characters). Accept = for == (minor slip). Do not accept changing `"X"` or the outputs.
4(b) J277 2.2.3	1 mark each to max 2: Convert the code / the first character to upper case (using `.upper`) ... ... before it is compared with `"X"` (on line 03) / after it is input (line 01) or when it is extracted (line 02) ... so `"x"` becomes `"X"` and the comparison is True Example: add `code = code.upper` after line 01 / `if code.length == 6 AND first.upper == "X" then`	**2**	Accept a description or code. "Use .upper" with no indication of what is converted or when = 0. Converting `"X"` in line 03 to upper case (it already is) = 0. An answer that compares with both `"X"` and `"x"` (no .upper) max 1.

5(a)
J277 2.2.3, 2.2.1
	1 mark each to max 5:
	Inputs the town name and the route number, storing each in a variable
	Takes the first three characters of the town name: `town.left(3)` / `town.substring(0, 3)`
	Converts (the first three characters of) the town name to upper case
	Concatenates the three parts in the correct order, with `"-"` between the letters and the route number
	Outputs the route code
	Example:
	`town = input("Enter the town name")`
	`route = input("Enter the route number")`
	`start = town.left(3)`
	`routeCode = start.upper + "-" + route`
	`print(routeCode)`
		5	Accept any variable names except reserved words (e.g. `input`). Converting the whole town name to upper case before slicing is fine. If the route number is input as an integer, do not penalise a missing `str()` when it is concatenated. `substring(1, 3)` or `[1:4]` loses the slicing mark. A comma in `print` (e.g. `print(start, "-", route)`) is not concatenation: no concatenation mark, but the output mark can be given. Building the code inside `print(...)` is fine for both marks. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M18

One-dimensional arrays: mark scheme

22 marks · OCR J277 spec 2.2.3 (the use of 1D arrays when solving problems; arrays as fixed length or static structures) and 2.1.2 (trace tables)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip is penalised once then follow through. Capitals in keywords are ignored. Python lists are accepted in place of arrays (e.g. `tickets = [0] * 7`). Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	Every element in an array must be the same data type Once an array has been declared, it cannot change size while the program runs	**2**	1 mark per correct tick. More than two ticks: deduct 1 mark for each extra tick (minimum 0).
1(b) J277 2.2.3	Max 1: One identifier/name holds all 50 values / no need to declare (and name) 50 separate variables A loop can use a variable as the index to process each value in turn / the same code can be repeated for every value The code is shorter / easier to write, read or maintain because the values are grouped under one name	**1**	"Easier", "quicker" or "more efficient" alone, with no reason = 0. "Uses less memory" = 0 (not true in general). Credit any other valid reason.
2(a) J277 2.2.3	16	**1**	Do not accept 7 (1-indexed counting) or `parcels[3]`.
2(b) J277 2.2.3	`parcels[1] = 15`	**1**	Index must be 1. Accept `parcels[1] = parcels[1] + 15`. Do not accept `parcels[2] = 15`, `parcels[0] = 15`, or `15 = parcels[1]` (wrong way round). Do not accept a whole new array declaration.
3(a) J277 2.2.3	Max 2: `array guests[5]` creates indexes 0 to 4 / arrays are 0-indexed so the last index is 4 (so) index 5 does not exist / is outside the array / is out of range The array is fixed length / static, so a sixth element cannot be added	**2**	"There is no room 5" alone = 0. Do not credit answers about the string, quotes or the name. Accept "the array only has 5 elements, numbered 0 to 4" for 2.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr>
<td>4(a)
J277 2.2.3, 2.1.2</td>
<td>1 mark for each correct column:
<code>i</code>: 0, 1, 2, 3
<code>low</code>: 0, 1, 2
Output: 2 (once, at the end)
<table>
<tr><th>I</th><th>LOW</th><th>OUTPUT</th></tr>
<tr><td></td><td>0</td><td></td></tr>
<tr><td>0</td><td>1</td><td></td></tr>
<tr><td>1</td><td></td><td></td></tr>
<tr><td>2</td><td>2</td><td></td></tr>
<tr><td>3</td><td></td><td></td></tr>
<tr><td></td><td></td><td>2</td></tr>
</table>
</td>
<td>3</td>
<td>Values must be in the correct order. Ignore repeated values on later rows (e.g. 1 written again beside i = 1). Accept low = 0 on the same row as i = 0 followed by 1. Output written as <code>print(2)</code> or with quotes: do not award. Any value of i above 3, or low above 2: lose that column's mark.</td>
</tr>
<tr>
<td>4(b)
J277 2.2.3</td>
<td>Index 0: <code>60</code> and index 2: <code>35</code>
Index 1: <code>85</code> and index 3: <code>60</code> (unchanged)</td>
<td>2</td>
<td>Final array: 60, 85, 35, 60. 1 mark per bullet; both values in a bullet are needed for its mark.</td>
</tr>
<tr>
<td>5(a)
J277 2.2.3</td>
<td>1 mark each to max 4:
Declares an array/list called <code>tickets</code> with 7 elements
Count-controlled loop that runs 7 times (e.g. <code>for day = 0 to 6</code>)
Inputs the number of tickets inside the loop
Stores each input in <code>tickets[day]</code>, using the loop variable as the index
Example:
<pre>array tickets[7]
for day = 0 to 6
    tickets[day] = int(input("Enter tickets sold"))
next day</pre>
</td>
<td>4</td>
<td>Accept <code>for day = 0 to tickets.length - 1</code>, Python <code>for day in range(7)</code>. Accept any loop variable name. A loop of 1 to 7 used with the index: loop mark only if it runs 7 times; the store mark is lost if index 7 is used (outside the array). Accept an empty list with append inside a 7-times loop: max 3 (no declaration mark). Accept a condition-controlled loop with a correct counter. Do not penalise a missing <code>int()</code>.</td>
</tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE

6(a)
J277 2.2.3
	1 mark each to max 6:
	Inputs the gallery number and stores it in a variable
	Uses the input variable as the index: `visitors[gallery]`
	...
	... adds 1 to that element and stores the result back in the same element
	Attempt at a loop to visit the elements of the array
	Loop covers every index 0 to 5 exactly once (no index 6)
	Outputs `visitors[g]` inside the loop, using the loop variable as the index
	Example:
	`gallery = int(input("Enter gallery number (0 to 5)"))`
	`visitors[gallery] = visitors[gallery] + 1`
	`for g = 0 to 5`
	`    print(visitors[g])`
	`next g`
		6	Accept `visitors[gallery] += 1`. Accept `for g = 0 to visitors.length - 1`, Python `range(6)` / `range(len(visitors))`. A Python `for v in visitors: print(v)` loop gains the loop, every-index and output marks. Six separate print lines with no loop: max 3 (first three marks). Adding 1 to a variable but not storing it back in the array loses the third mark. The update placed inside the output loop, so it is repeated: lose the third mark. Do not penalise a missing `int()`. Do not accept `input` as a variable name. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M19

Two-dimensional arrays and looping through them: mark scheme

21 marks · OCR J277 spec 2.2.3 (the use of arrays when solving problems, including 1D and 2D arrays; arrays as fixed length or static structures; use of 2D arrays to emulate database tables of a collection of fields and records), with 2.2.1 (iteration)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. 2D indexes may be written `[r,c]` or `[r][c]`, but must be in the row, column order given in the question. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Accept Python `range(0, 5)` / `range(5)` for `0 to 4`. Do not accept flowcharts, structured English or bullet points for write tasks.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	Every element must be of the same data type: True `grid[4,6]` can be used: False Extra rows can be added while running: False	**3**	1 mark per correct row. Both boxes ticked in a row: 0 for that row. Row 2: array `grid[4,6]` gives indexes 0 to 3 and 0 to 5, so `grid[4,6]` is outside the array. Row 3: arrays are fixed length / static.
2(a) J277 2.2.3	`Haddad`	**1**	Accept with or without quotation marks. Do not accept "Yoga" or "Lindqvist".
2(b) J277 2.2.3	`instructors[4,2] = "Yoga"`	**1**	Accept `instructors[4][2] = "Yoga"`. Indexes must be 4 then 2. Allow missing quotes (minor slip). Do not accept ==, or the value on the left (`"Yoga" = instructors[4,2]`).

2(c) J277 2.2.3	1 mark each to max 3:
	Loop over every record / row 0 to 4
	Checks the activity field of the current row: `instructors[r,2] == "Climbing"`
	Outputs the surname field of the same row: `print(instructors[r,1])` (inside the IF)
	Example:
	`for r = 0 to 4`
	`  if instructors[r,2] == "Climbing" then`
	`  print(instructors[r,1])`
	`  endif`
	`next r`
		3	The check and output marks need the loop variable as the first (row) index. Accept a while loop with a counter, and `for r = 0 to instructors.length - 1` or `len(instructors)`. Loop 1 to 5 or 0 to 5: loop mark lost, FT for the other two marks. Hard-coded checks of rows 1 and 3 with no loop: 0 (iteration is required). Output of the whole record or the ID instead of the surname: lose the output mark.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.2.3	177	**1**	Working: 55 + 49 + 44 + 29 = 177 (row index 1, every reading).

3(b)
J277 2.2.3
2.1.2
	1 mark per correct line:
	`0 2`
	`1 3`
	`2 1`
		3	Values below 30 are 27 (bed 0, reading 2), 29 (bed 1, reading 3) and 26 (bed 2, reading 1); the outer loop works through the beds in order. Ignore spacing (accept `02`). Do not accept the moisture values (27, 29, 26) instead of the indexes. All three correct but in the wrong order: max 2. Any extra line output: max 2.

3(c)
J277 2.2.3
	1 mark each to max 3:
	Loop over the three beds (0 to 2) adding to a running total that starts at 0
	Adds the correct element each time: `moisture[bed,3]`
	Divides the total by 3 (after the loop) and outputs the result
	Example:
	`total = 0`
	`for bed = 0 to 2`
	`    total = total + moisture[bed,3]`
	`next bed`
	`print(total / 3)`
		3	The result is 100 / 3 = 33.3 (recurring); candidates do not need to state it. Accept `moisture[bed][3]`. `moisture[3,bed]` (indexes reversed): lose the second mark. Dividing inside the loop and outputting once correctly: allow. Dividing by 4: lose the third mark. No loop (three values added directly): max 1 (third mark only). Do not penalise rounding the output.

4(a)
J277 2.2.3
2.2.1
	1 mark each to max 6:
	Counter set to 0 before the loops
	Outer loop over the rows 0 to 4 ...
	... with an inner loop over the lockers 0 to 7 nested inside it
	Checks `rented[row, locker]` using both loop variables, row first, for a free locker (`== False` / `NOT`) ...
	... and adds 1 to the counter when it is free
	Outputs the counter once, after both loops have finished
	Outputs `All lockers rented` only when the count is 0
	Example:
	`free = 0`
	`for row = 0 to 4`
	`  for locker = 0 to 7`
	`  if rented[row, locker] == False then`
	`  free = free + 1`
	`  endif`
	`  next locker`
	`next row`
	`print(free)`
	`if free == 0 then`
	`  print("All lockers rented")`
	`endif`
		6	Accept `if NOT rented[row, locker]` / `!= True`. Accept counting rented lockers and outputting `40 - count` (then "All lockers rented" when 40 are rented). Accept loops that go 1 to 5 / 1 to 8 only if the index is adjusted (`row - 1`); otherwise lose the loop mark it affects. Indexes reversed (`rented[locker, row]`): lose the check mark. Output inside the loops (prints many times): lose the output mark. Accept a flag instead of the count for the last mark. Accept the words `All lockers rented` with any minor change in capitals. Do not accept `input` as a variable name.

Total: 21 marks

MODULE M PROGRAMMING · M21

File handling: open, read, write and close: mark scheme

22 marks · OCR J277 spec 2.2.3 (the use of basic file handling operations: open, read, write, close), with 2.2.1 (iteration) and 2.1.2 (identifying errors)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative words; **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it depends on the bullet before it. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Penalise a repeated syntax slip once, then follow through.

QUESTION	ANSWER	MARKS	GUIDANCE
1 J277 2.2.3	Adds a line of text from the program to the end of the file: Write Releases the file once the program has finished using it: Close Connects the program to a named file so that it can be used: Open Copies the next line of the file into the program ...: Read	**4**	1 mark per correct row. A row with more than one tick scores 0 for that row.
2(a) J277 2.2.3	CONTENTS OF TICKETS.TXT T101 T102 T103 T104 T101 and T102 kept, at the start, in this order T103 then T104 added after them, in this order	**2**	`writeLine()` adds each line to the END of the file, so the existing lines are not lost. Ignore quotation marks round the codes and ignore empty boxes. Max 1 if any extra code is shown (e.g. a code repeated). An answer showing only T103 and T104 scores 0.
2(b) J277 2.2.3	Creates the file: `newFile("archive.txt")` Opens the file and stores it in a variable: e.g. `archiveFile = open("archive.txt")`	**2**	Any sensible variable name. The opening mark needs the file name and a variable. Allow single quotes. In Python, `archiveFile = open("archive.txt", "w")` (or `"a"` / `"x"`) both creates and opens the file: award both marks. Python `open("archive.txt")` or `"r"` mode alone (read only, does not create) scores the opening mark only. The create mark must come before (or with) the open.
3(a) J277 2.2.3 J277 2.2.1	15.25	**1**	Correct answer only (61 ÷ 4). Do not accept 61, 15 or 15.3. Do not accept `total / count`.

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) J277 2.2.3	4 // four (times)	**1**	Once for each line in the file. Do not accept 5.
3(c) J277 2.2.3 J277 2.2.1	It starts a loop // repeats lines 05 to 07 // repeats reading a line (and adding it to the total) ... until the end of the file is reached // while there are still lines left to read // so every line / reading in the file is read It checks whether the end of the file has been reached (before each line is read) // stops the program trying to read past the last line	**2**	Max 2. Credit any other valid description of line 04. Do not accept "it reads the file" alone (line 05 reads). Do not accept "it ends the file" or "it closes the file".
4 J277 2.2.3 J277 2.1.2	Line 01 ... `setFile = open("settings.txt")` Line 02 ... `volume = setFile.readLine()`	**4**	Each correction mark depends on the matching line number. Line 01: the file name must be in quotes; allow single quotes; allow Python `open("settings.txt", "r")`. Line 02: the program must read, not write; allow Python `setFile.readline()`. Do not accept `readLine` without the variable assignment. Do not credit lines 03 or 04 as errors.

5
J277 2.2.3
J277 2.2.1
	Opens `orders.txt` (stored as a variable / file handle) before the orders are written
	Attempt at iteration (any loop)
	Order input inside the loop (so a new order is asked for each time)
	Loop stops when `stop` is entered, and `stop` is not written to the file
	Each order written to the file (e.g. `writeLine(order)`) inside the loop
	File closed after the loop has ended
	Number of orders counted (incremented once per order written) and output after the loop
	Example answer (ERL):
	`orderFile = open("orders.txt")`
	`count = 0`
	`order = input("Enter order")`
	`while order != "stop"`
	`  orderFile.writeLine(order)`
	`  count = count + 1`
	`  order = input("Enter order")`
	`endwhile`
	`orderFile.close()`
	`print("Orders saved: " + str(count))`
		6	1 mark per bullet, max 6. Accept any write or append mode in Python (`"w"` or `"a"`), since the file starts empty; accept `.write(order + "\n")` or `.write(order)` for writing. A Python `with open(...)` block closes the file: award the close mark if the writes are inside the block. Do-until / `while True` with `break` loops are fine if `stop` is not written. Do not award the stop mark if `stop` is written to the file or the loop never ends. Do not award the close mark if the file is closed inside the loop (unless it is reopened each time). Allow a message with the count, or the count alone. Do not accept `input` used as a variable name.

Total: 22 marks

MODULE M PROGRAMMING · M22

Procedures: mark scheme

22 marks · OCR J277 spec 2.2.3 (the use of procedures; how to use sub programs to produce structured code), 2.3.1 (maintainability: use of sub programs), 2.1.2 (refining algorithms)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers; **//** separates alternative statements for the same mark. Words in (brackets) are not needed for the mark. **...** at the end of one point and the start of the next means the second depends on the first. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip is penalised once then follow through. Capitals in keywords are ignored. Credit any other valid answer.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	Must have at least one parameter: **False** Can be called more than once: **True** Code only runs when the procedure is called: **True**	**3**	1 mark per correct row. Two ticks in a row: 0 for that row. Row 1: procedures can be written with or without parameters, e.g. `procedure agePass()` in the J277 spec.
1(b) J277 2.2.3	A function returns a value (to where it was called) but a procedure does not // a procedure does not return a value but a function does	**1**	Max 1. Accept "a function call can be used in an expression / assigned to a variable, but a procedure call cannot". NE: "a function does a calculation", "a procedure does a task" (both can). Do not accept "a procedure has no parameters" (wrong).

2(a) J277 2.2.3, 2.1.2	1 mark each to max 4:
	`----------` then `Spring Craft Fair` as the first two lines, in that order
	`Ticket 1` followed by `----------`
	`Ticket 2` followed by `----------`
	`Enjoy your visit` as the last line, output once only
	Full output:
	`----------`
	`Spring Craft Fair`
	`Ticket 1`
	`----------`
	`Ticket 2`
	`----------`
	`Enjoy your visit`
		4	Ignore the number of dashes. Ignore quotation marks round the output. Do not award a mark if the line it depends on is missing or in the wrong order (e.g. `Spring Craft Fair` before the first line of dashes = 0 for the first point). Output that includes `print(...)` or the code itself: penalise once.

QUESTION	ANSWER	MARKS	GUIDANCE
2(b) J277 2.2.3	1 mark each to max 2: The procedure `printHeader()` is called // the program jumps to (line 04,) `printHeader()` The code inside the procedure (lines 05 to 06) runs // it calls `printLine()` and then outputs `Spring Craft Fair` When the procedure ends, the program returns to / carries on from the line after the call (line 09)	**2**	NE: "it prints the header" alone (no idea of calling). Do not accept "the procedure is defined / created on line 08". Do not accept "the program returns to line 08 and runs it again" for the third point.

3(a)
J277 2.2.3
	1 mark each to max 3:
	Procedure header with the identifier `showMenu`: `procedure showMenu()` / `def showMenu():`
	Outputs `1 Borrow`, `2 Return` and `3 Renew`, in this order, inside the procedure
	End of the procedure shown: `endprocedure` / end of the indented block / `End Sub`
	Example:
	`procedure showMenu()`
	`  print("1 Borrow")`
	`  print("2 Return")`
	`  print("3 Renew")`
	`endprocedure`
		3	Accept the three lines as one output with line breaks (e.g. \n). Ignore small differences in spacing/case of the menu text. Missing quotation marks round the text: penalise once. Do not award the header mark for `function showMenu()` or a header with a parameter that is not used. A procedure that also returns a value: max 2. Python `def` with the body indented gets the end mark.

3(b) J277 2.2.3	`showMenu()`	**1**	Accept `call showMenu()` / `Call showMenu()` (VB). BOD `showMenu` without brackets. Do not accept the header `procedure showMenu()`, or `x = showMenu()` (a procedure does not return a value).

QUESTION	ANSWER	MARKS	GUIDANCE

4(a)
J277 2.2.3, 2.1.2
	1 mark each to max 4:
	Procedure `openGate()` defined, with an end (`procedure openGate()` ... `endprocedure` / `def openGate():` with indented body)
	... containing the three repeated outputs (`Gate opening`, `Please walk through`, `Gate closing`) in this order, and no other outputs
	`openGate()` called in the CARD branch **and** in the CODE branch (after the accepted message), replacing the repeated lines
	Rest of the program kept: input, selection, `Card accepted` / `Code accepted` / `Please ask at reception`, and the gate is **not** opened for any other input
	Example:
	`procedure openGate()`
	`    print("Gate opening")`
	`    print("Please walk through")`
	`    print("Gate closing")`
	`endprocedure`
	
	`method = input("Enter CARD or CODE")`
	`if method == "CARD" then`
	`    print("Card accepted")`
	`    openGate()`
	`elseif method == "CODE" then`
	`    print("Code accepted")`
	`    openGate()`
	`else`
	`    print("Please ask at reception")`
	`endif`
		4	The procedure may be written before or after the main program. Accept any logically equivalent structure, e.g. `if method == "CARD" OR method == "CODE"` with a nested `if` for the accepted message and one call to `openGate()`, as long as the output is the same for every input. Call made in only one branch: do not award the third mark. Call placed before the accepted message (changes the order of output): do not award the third mark. `openGate()` called in the `else` branch, or after `endif` for every input: do not award the fourth mark. Do not accept flowcharts, structured English or bullet points.

4(b) J277 2.3.1, 2.2.3	1 mark for each benefit, 1 mark for an explanation of it in this program, two benefits, max 4: The three lines are written only once ... // less repeated code ... ... so the program is shorter / quicker to write (the gate code is not typed out twice) Easier to maintain / update ... ... e.g. if a gate message changes it only needs changing in one place // both branches always give the same gate messages / no chance of changing one copy and forgetting the other Easier to test / debug ... ... the gate code can be tested once on its own // an error in the gate code only needs fixing in one place Easier to read / understand / more structured ... ... the name `openGate()` shows what that block of code does // the main program is shorter and easier to follow Can be reused ... ... e.g. called again if another way of entering is added (such as a phone pass) / used in other programs	**4**	Mark in pairs: the explanation mark needs its matching benefit. Max 2 if no answer refers to this program (the gate, its messages, the two branches). Do not award the same idea twice (e.g. "shorter" and "less code"). NE: "faster", "more efficient", "uses less memory" on their own. Do not accept "the program runs faster" (it does not).

Total: 22 marks

MODULE M PROGRAMMING · M23

Functions and return values: mark scheme

22 marks · OCR J277 spec 2.2.3 (the use of functions and procedures; sub programs), with 2.2.1 (MOD, DIV, selection) used in context

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. **...** means the later mark depends on the earlier one. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip (e.g. == for assignment) is penalised once then follow through. Capitals in keywords are ignored. Do not accept maths symbols (×, ≥, ≤) or English ("is more than") in place of operators.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	Named sub program, defined once, called many times: Both Uses `return` to send a value back: Function Carries out a task but does not return a value: Procedure	**3**	1 mark per correct row. More than one tick in a row: 0 for that row.
2(a) J277 2.2.3	04 / 4	**1**	Accept `return cost` written out instead of the number.
2(b) J277 2.2.3	60	**1**	Working: 3 × 4 = 12 pages; 12 × 5 = 60. Accept 60p. Do not accept £60 or 0.6 on its own (the function returns pence).
2(c) J277 2.2.3	Two calls with the arguments in the correct order: `printCost(2, 5)` and `printCost(4, 3)` ... ... results added and assigned to `orderTotal` Example: `orderTotal = printCost(2, 5) + printCost(4, 3)`	**2**	(Value 50 + 60 = 110.) Arguments swapped in one or both calls (`printCost(5, 2)`) gives the same value here but does not match the parameters (pages, copies): do not award MP1, allow MP2 FT. Do not accept `orderTotal = 110` (no function used) or two statements (the question asks for one). Do not accept == for assignment.
3(a) J277 2.2.3 / 2.2.1	`boxesNeeded(16)`: 2 `boxesNeeded(17)`: 3 `boxesNeeded(5)`: 1	**3**	1 mark per correct row. Working: 16 DIV 8 = 2, 16 MOD 8 = 0 so no extra box; 17 DIV 8 = 2, 17 MOD 8 = 1 so 2 + 1 = 3; 5 DIV 8 = 0, 5 MOD 8 = 5 so 0 + 1 = 1.
3(b) J277 2.2.3	20: `Use a trolley` 16: `Carry by hand`	**2**	Working: `boxesNeeded(20)` returns 3 and 3 > 2 is True; `boxesNeeded(16)` returns 2 and 2 > 2 is False. Ignore quotation marks. Do not accept the number of boxes (3 / 2) as the output.

QUESTION	ANSWER	MARKS	GUIDANCE
4(a) J277 2.2.3	Line 04 (identified) ... ... `return mean` Line 06 (identified) ... ... `result = meanReading(4, 9, 8)`	**4**	The correction mark for each error needs the right line number. Line 04: accept `return total / 3`; do not accept any line that still prints instead of returning. Line 06: the call must be kept and its returned value stored in `result`. Do not accept == for assignment. Do not accept changing line 07 only (e.g. calling the function inside the print): the program must store the mean in `result`. Do not accept changes to lines 01 to 03.

5(a)
J277 2.2.3 / 2.2.1
	1 mark each to max 6:
	Function defined with a parameter (e.g. `function isComfortable(temp)` ... `endfunction` / `def isComfortable(temp):`)
	Correct check of both limits inside the function: `temp >= 18 AND temp <= 24`
	Returns `True` when comfortable and `False` otherwise (must use return)
	Inputs the room temperature in the main program and stores it in a variable
	Calls `isComfortable()` with the entered temperature as the argument ...
	... uses the returned value in selection to output `Comfortable` and `Adjust heating` in the correct cases
	Example:
	`function isComfortable(temp)`
	`    if temp >= 18 AND temp <= 24 then`
	`        return True`
	`    else`
	`        return False`
	`    endif`
	`endfunction`
	
	`roomTemp = float(input("Enter the room temperature"))`
	`if isComfortable(roomTemp) == True then`
	`    print("Comfortable")`
	`else`
	`    print("Adjust heating")`
	`endif`
		6	Accept `return temp >= 18 AND temp <= 24` for MP2 and MP3. Temperatures can include decimals, so `temp > 17` or `temp < 25` lose MP2, as do `temp > 18` or `temp < 24`. Each side must be a full comparison (`temp >= 18 AND <= 24` loses MP2). Accept `if isComfortable(roomTemp) then` (no `== True`), or storing the returned value in a variable first. Do not penalise missing or different casting of the input. MP1: a parameter that is overwritten by an `input` inside the function does not get MP1. MP3: printing inside the function instead of returning does not get MP3. Accept the function written after the main program. Accept any variable names except reserved words (e.g. `input`). Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE M PROGRAMMING · M24

Parameters and arguments: mark scheme

20 marks · OCR J277 spec 2.2.3 (the use of functions and procedures; how to use sub programs to produce structured code; arrays passed into sub programs) and 2.3.1 (maintainability: use of sub programs)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip is penalised once then follow through. Capitals in keywords are ignored. Students do not need to use the word "argument" anywhere: "the value passed (to the parameter)" is equally creditworthy.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	`bill` and `percent`	**1**	Both needed for the mark. Do not accept `40` and `10` (these are the values passed in the call). Do not accept `extra` or `total` (variables, not parameters).
1(b) J277 2.2.3	(Line) 06	**1**	Accept 6. Do not accept 01 (that is where the function is defined).
1(c) J277 2.2.3	10	**1**	The second value in the call goes to the second parameter.
1(d) J277 2.2.3	44	**1**	Accept 44.0. Working: extra = 40 × 10 / 100 = 4; 40 + 4 = 44. Do not accept 4 or 440.
2(a) J277 2.2.3	Used in the same way as a variable: True Can only ever have one parameter: False Values matched to parameters in the order written: True	**3**	1 mark per correct row. Both boxes ticked in a row: 0 for that row.
3(a) J277 2.2.3	Line 09: `bolts low: order 8` Line 10: `nuts OK` Line 11: `washers OK`	**3**	1 mark per correct row. Ignore quotation marks and minor spacing. Line 09: 20 – 12 = 8. Line 11: 15 < 15 is False, so the `else` branch runs; `washers low: order 0` = 0 marks for that row.

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) J277 2.2.3	1 mark each to max 2: The values are matched to the parameters by position / in order ... ... so `item` gets 20 / `quantity` gets `"bolts"` / `minimum` gets 12 / the values go into the wrong parameters A string (`"bolts"`) would be compared with a number on line 02 / the data types do not match what the procedure expects / it would cause an error (or wrong output)	**2**	"The values are in the wrong order" alone = 1. For 2 the answer must say what goes wrong as a result (wrong parameter gets the value, data type mismatch or error). Accept "should be `stockMessage("bolts", 12, 20)`" as showing the order point (1). Credit any other valid answer.
4(a) J277 2.2.3 / 2.3.1	1 mark each to max 2: The code is written only once / less code to write / shorter program The same procedure is reused for every room by passing a different room number Easier to maintain / a change to the sign only has to be made in one place Fewer chances of errors / only one piece of code to test / easier to debug Easier to read / understand / more structured program	**2**	Two different benefits needed. "Quicker" or "more efficient" alone = NE (say what is quicker, e.g. quicker to write). Credit any other valid answer.

5(a)
J277 2.2.3
	1 mark each to max 4:
	A counter is set to 0 (before the loop)
	A loop that visits every element of `readings` (e.g. `for i = 0 to readings.length - 1` / for each value)
	Each element is compared with the parameter `limit` using >, and the counter is increased by 1 when it is greater
	The count is returned (after the loop has finished)
	Example:
	`function countAbove(readings, limit)`
	`    count = 0`
	`    for i = 0 to readings.length - 1`
	`        if readings[i] > limit then`
	`            count = count + 1`
	`        endif`
	`    next i`
	`    return count`
	`endfunction`
		4	Must use the parameters: a fixed value (e.g. `> 30`) in place of `limit` loses the comparison mark, and inputting a new value into `limit` or `readings` (overwriting the parameter) loses the mark it affects. >= loses the comparison mark. Accept a loop to 6 / `range(7)` or `range(len(readings))` (a week has 7 values); allow an off-by-one bound explained by Python `range`. Return inside the loop loses the return mark. Printing the count instead of returning it loses the return mark. Do not accept flowcharts, structured English or bullet points.

5(b) J277 2.2.3	`countAbove(dayTemps, 30)`: function called with the array and 30, in that order The returned value is stored in `hotDays` (`hotDays = ...`) Example: `hotDays = countAbove(dayTemps, 30)`	**2**	`countAbove(30, dayTemps)` (wrong order) loses the first mark. Allow follow through if 5(a) used different parameter names. Do not accept `hotDays == countAbove(...)` (comparison, not assignment) for the second mark. A call on its own with no assignment = max 1.

Total: 20 marks

MODULE M PROGRAMMING · M25

Local and global variables: mark scheme

22 marks · OCR J277 spec 2.2.3 (the use of local variables/constants and global variables/constants within functions and procedures; the use of procedures)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. **...** at the end of one bullet and the start of the next means the second mark depends on the first. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip is penalised once then follow through. Credit any other valid answer.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	Used in the main program and every sub program: Global variable Declared inside a procedure or function: Local variable Only exists while its sub program is running: Local variable	**3**	1 mark per correct row. Two ticks in a row: 0 for that row.
2(a) J277 2.2.3	`totalTakings`	**1**	Correct answer only. Ignore case. Do not accept `global` on its own (the keyword, not the variable).
2(b) J277 2.2.3	`RATE`	**1**	Correct answer only. Ignore case. Accept "RATE = 2" / line 04. Do not accept `fee` or `hours` (local variables, not a constant).

2(c) J277 2.2.3	`Fee: 6`
	`Fee: 10`
	`Total: 16`
		3	1 mark per correct line, in this order. Ignore quotation marks and small spacing differences. A line given as the number only (e.g. 6, 10, 16) = max 2 in total. Working: 3 × 2 = 6; 5 × 2 = 10; 0 + 6 + 10 = 16.

2(d) J277 2.2.3	1 mark each to max 2: `fee` is a local variable / is declared inside the procedure `takePayment` ... ... so it can only be used / accessed inside that procedure // cannot be used in the main program ... `fee` only exists while the procedure is running / no longer exists after the procedure ends (line 08)	**2**	Second and third points need `fee` to be identified as local (or declared in the procedure). "fee has not been declared" alone = NE. "It is a syntax error" alone = 0.
2(e) J277 2.2.3	1 mark each to max 2: It is changed inside the procedure **and** used in the main program (line 12) / it is needed in more than one part of the program It must keep its value between calls to the procedure / keeps adding up across both payments A local variable would be lost / reset when the procedure ends (so the total could not be kept or printed)	**2**	Answers must refer to this program for the first two points. "So it can be used anywhere" with no link to this program = max 1. Do not accept "because it is a total" alone.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.2.3	1 mark for each benefit and 1 for its explanation, max 2 benefits (max 4): A local variable can only be changed inside its own sub program ... ... so other programmers' code cannot accidentally change its value / fewer unexpected side effects The same variable name can be used in different sub programs ... ... without the variables clashing / programmers do not need to check which names others have used Each sub program is self-contained / does not depend on variables outside it ... ... so it can be tested / debugged on its own // can be reused in another program more easily Local variables only exist while the sub program runs ... ... so the memory they use is freed when the sub program ends	**4**	Mark in pairs: an explanation only scores with its benefit. The same benefit given twice scores once. "Easier to debug" with no reason = NE for the explanation mark unless linked to a benefit. Do not accept "faster" or "more secure" alone. Credit any other valid benefit with an explanation.

4(a)
J277 2.2.3
	1 mark each to max 5:
	Procedure defined with the name `addStamps`, a parameter in brackets and an `endprocedure` (or equivalent end of block)
	Stamps worked out as amount `DIV 5` / `int(amount / 5)` / `amount // 5` (or any method that gives the whole number of full £5s) ...
	... and stored in a variable declared inside the procedure (local)
	`totalStamps = totalStamps + stamps` (global updated by adding, not replaced)
	Outputs the number of stamps earned (the local variable)
	Example:
	`procedure addStamps(spend)`
	`    stamps = spend DIV 5`
	`    totalStamps = totalStamps + stamps`
	`    print(stamps)`
	`endprocedure`
		5	Accept a function in place of a procedure. Python: accept `def addStamps(spend):` and accept with or without a `global totalStamps` line inside. Do not award the "stored ... local" mark if the stamps are stored in `totalStamps` or in a variable made global. Setting `totalStamps = 0` inside the procedure loses the update mark. `spend / 5` with no rounding down does not get the calculation mark. A parameter immediately overwritten by an input loses the first mark. Do not accept flowcharts, structured English or bullet points.

4(b) J277 2.2.3	6	**1**	Correct answer only. Working: 23 DIV 5 = 4; 12 DIV 5 = 2; 0 + 4 + 2 = 6.

Total: 22 marks

MODULE M PROGRAMMING · M26

Random numbers: mark scheme

20 marks · OCR J277 spec 2.2.3 (random number generation: create and use random numbers in a program), with 2.2.2 (data types), 2.2.3 (arrays), 2.2.1 (selection, iteration), 2.1.2 (trace tables)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. **...** means the second point depends on the first. Code answers are marked for logic, not exact syntax: any high-level language or OCR Exam Reference Language is accepted (e.g. Python `random.randint(1, 20)` for `random(1, 20)`); penalise a repeated syntax slip once, then follow through. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3, 2.2.2	• `seat`: integer • `seat`: smallest 1 and largest 40 • `wind`: real • `wind`: smallest −3.0 and largest 3.0	**4**	1 mark per bullet. Accept float for real; accept int for integer. Both values needed for each range mark; accept −3 and 3. Do not accept 2 or 39 (random is inclusive at both ends), or 0 as the smallest `wind` value.
2(a) J277 2.2.3	`code = random(100, 999)`	**1**	Must be stored in `code`, with 100 and 999 as the limits. Accept a high-level language equivalent, e.g. `code = random.randint(100, 999)`. Do not accept `random(100, 1000)` in ERL or `randint(100, 1000)` in Python. Accept `randrange(100, 1000)` (Python's upper limit is excluded).
2(b) J277 2.2.3	`pause = random(0.5, 2.5)`	**1**	Must be stored in `pause`, with 0.5 and 2.5 as the limits. Accept `random.uniform(0.5, 2.5)` or another correct high-level method. Do not accept `random(1, 2)` or any version that gives whole numbers only.
3(a) J277 2.2.3	1 mark each to max 2: • The fact at index 0 / the first fact is never shown • When 6 is chosen there is no element at index 6 / the index is outside the array ... so the program crashes / gives an error / cannot print a fact	**2**	"It picks the wrong fact" alone: not enough. Accept "index out of range" for the second bullet. Do not accept "it will not be random".

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(b)
J277 2.2.3</td><td><code>pick = random(0, 5)</code></td><td>1</td><td>Accept <code>pick = random(0, facts.length - 1)</code>. Accept <code>pick = random(1, 6) - 1</code>. Do not accept a change to line 02 only.</td></tr>
<tr><td>4(a)
J277 2.2.3, 2.1.2</td><td>1 mark per correct column:
• growth: 3, 4, 1, 2
• height: (5,) 8, 12, 13, 15
• days: (0,) 1, 2, 3, 4
• Output: Ready after 4 days (once, at the end)
<table><tr><th>GROWTH</th><th>HEIGHT</th><th>DAYS</th><th>OUTPUT</th></tr><tr><td></td><td>5</td><td>0</td><td></td></tr><tr><td>3</td><td>8</td><td>1</td><td></td></tr><tr><td>4</td><td>12</td><td>2</td><td></td></tr><tr><td>1</td><td>13</td><td>3</td><td></td></tr><tr><td>2</td><td>15</td><td>4</td><td>Ready after 4 days</td></tr></table></td><td>4</td><td>Values must be in the order shown; ignore which row each value is on and repeated values. Output: ignore quotation marks and small spacing slips; FT the candidate's final <code>days</code> value. Do not award the Output mark if an output is written on every row.</td></tr>
<tr><td>4(b)
J277 2.2.3</td><td>3</td><td>1</td><td>Correct answer only. The largest growth is 4 a day: 5 + 4 + 4 = 13 (not enough), 13 + 4 = 17. Do not accept 1, 2 or 10.</td></tr>
</table>

5(a)
J277 2.2.3, 2.2.1
	1 mark each to max 6:
	• Asks for / inputs the number of customers and stores it (as an integer)
	• Loop that runs once for each customer (uses the number input)
	• Generates a random whole number from 1 to 20 inside the loop and stores / uses it
	• Checks whether the number is 20 and outputs "Free drink" ...
	• ... otherwise outputs the number
	• Counts the free drinks (count set to 0 before the loop and increased by 1 only for a 20) ...
	• ... and outputs the count after the loop
	
	Example answer (ERL):
	`customers = int(input("Number of customers: "))`
	`freeDrinks = 0`
	`for c = 1 to customers`
	`  number = random(1, 20)`
	`  if number == 20 then`
	`  print("Free drink")`
	`  freeDrinks = freeDrinks + 1`
	`  else`
	`  print(number)`
	`  endif`
	`next c`
	`print("Free drinks won: " + str(freeDrinks))`
		6	Allow any logically valid method, e.g. a while loop with a counter. Random number generated once, before the loop: do not award bullet 3. `random(1, 21)` in ERL / `randint(1, 21)` in Python: do not award bullet 3. Count output inside the loop: do not award the last bullet. Accept `>= 20` for `== 20`. Missing `int()` / casting: do not penalise. Flowcharts or written English: 0.

Total: 20 marks

MODULE M PROGRAMMING · M27

Records: mark scheme

20 marks · OCR J277 spec 2.2.3 (the use of records to store data; arrays as fixed length or static structures; use of 2D arrays to emulate database tables of a collection of fields, and records), 2.2.2 (choosing suitable data types for data in a given scenario), with 2.2.1 and 2.1.2 for Q3 and Q4(d)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers: mark the logic, not the language. Allow OCR Exam Reference Language or any high-level language (for example a Python class, dictionary or list, or a record/structure type in another language), and allow any logically valid method. Do not penalise minor syntax slips (missing casting, capitals, a missing colon, and for AND). A recurring slip: penalise once, then follow through. Do not accept maths symbols such as ≥ or ≤ in code. Outputs: ignore quotation marks round output text.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	1 mark each to max 2: A data structure / a way of storing data (under one identifier) ... that groups together (a collection of) related fields / data items / values about one item / thing / entity (e.g. one tool) The fields can be of different data types Each field has its own (field) name / is accessed by its name	**2**	Allow "a row in a table" / "one row of data about one item" for 1 mark. An example alone (e.g. "TH204, Pressure washer") max 1. Do not accept a definition of a field. Credit any other valid point.
1(b) J277 2.2.3	One item of data / one piece of information / one attribute within a record (e.g. `dailyRate`)	**1**	Allow "a column in a table / of a 2D array" / "one category of data stored for every record". Do not accept an example on its own. Do not accept "a row".
1(c) J277 2.2.3	1 mark each to max 2: A record's fields can be different data types // an array stores only one data type A field is accessed by its (field) name // an array element is accessed by its index / position (number) A record holds related data about one item // a 1D array holds a list of values of the same kind (e.g. many descriptions)	**2**	Each difference must be clear for one of the two; the other side may be implied (e.g. "an array can only hold one data type" = 1 mark). Do not accept "an array is fixed length" alone (a record's fields are also fixed). Do not accept "a record is bigger" or "an array is faster". Credit any other valid difference.
2 J277 2.2.2, 2.2.3	1 mark per correct row: `description`: string `dailyRate`: real / float `minDays`: integer `available`: Boolean	**4**	Accept: str; float / decimal / single / double for real; int; bool. Accept text / varchar for string. Do not accept: number / numeric on their own; currency; character / char for `description`; yes-no / true-false for Boolean; string for `dailyRate` or `minDays`.

QUESTION	ANSWER	MARKS	GUIDANCE

3(a)
J277 2.2.3, 2.2.1
	1 mark per line:
	`Pressure washer`
	`Cost: 72.0`
		2	Accept `72` or `72.00` in place of `72.0`. Allow the second line with or without "Cost: ". Ignore the order only if both lines are clearly given as separate outputs. Do not accept `Cannot hire` as well. Do not accept `hireTool.description` or `print(...)` as an output.

3(b) J277 2.2.3, 2.2.1	`Cannot hire`	**1**	Do not accept any other output with it.
3(c) J277 2.2.3	Assigns to the `dailyRate` field of `hireTool` its current value plus 1.50 `hireTool.dailyRate = hireTool.dailyRate + 1.50`	**1**	Allow `hireTool.dailyRate += 1.5`. Allow `hireTool.dailyRate = 25.50`. Allow the field accessed in another language's style, e.g. `hireTool["dailyRate"]`. Do not accept `dailyRate = dailyRate + 1.50` (record not named), `hireTool = hireTool + 1.50`, or `=+`.
4(a) J277 2.2.3	`Tile cutter`	**1**	Ignore quotation marks.
4(b) J277 2.2.3	All five values of row 3: `TH427`, `Leaf blower`, `9.75`, `1`, `True`	**1**	All five values needed (a whole row). Ignore order, quotation marks and separators. Allow field names with the data. Do not accept a single value or the column of IDs.
4(c) J277 2.2.3	Assigns "True" to row 0, column 4 `tools[0, 4] = "True"`	**1**	Allow `True` without quotation marks. Allow `tools[0][4]`. Do not accept `tools[4, 0]`, `tools[1, 5]` or `==`.

QUESTION	ANSWER	MARKS	GUIDANCE

4(d)
J277 2.2.3, 2.1.2
	1 mark each to max 4:
	Iterates through all five records / rows (0 to 4)
	Checks the `available` field / column 4 is "True"
	Checks the `dailyRate` field / column 2 is less than 20 (`< 20`) ...
	... and outputs the description / column 1 only when **both** conditions are true
	`for row = 0 to 4`
	`    if tools[row, 4] == "True" AND float(tools[row, 2]) < 20.00 then`
	`        print(tools[row, 1])`
	`    endif`
	`next row`
	This outputs `Tile cutter` and `Leaf blower`.
		4	Allow nested `if` statements instead of AND. Allow `True` without quotation marks and missing casting. Do not accept `<= 20` for mark point 3. Using OR instead of AND: do not award mark point 4. Allow `for row in range(5)` or a `while` loop that ends after row 4; a loop of 1 to 5 does not get mark point 1. Allow `tools[row][1]` style. Five separate `if` statements with fixed rows instead of a loop: do not award mark point 1 (other mark points may be awarded). Outputting the whole record instead of the description: do not award mark point 4.

Total: 20 marks

MODULE M PROGRAMMING · M28

Writing maintainable code: mark scheme

20 marks · OCR J277 spec 2.3.1 (maintainability: use of sub programs, naming conventions, indentation, commenting; why commenting is useful and applying it), 2.2.3 (the use of sub programs: functions)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers: mark the logic, not the language. Allow OCR Exam Reference Language or any high-level language, and allow any logically valid method. Do not penalise minor syntax slips (capitals, a missing colon, # instead of // for a comment). A recurring slip: penalise once, then follow through. Do not accept maths symbols such as ≥ or ≤ in code. Explain questions: a generic answer that does not refer to the program given scores max 1 where stated.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.1	1 mark each: Adding comments that explain what each section of code does Indenting the statements inside each loop and `if` statement	**2**	More than two ticks: deduct 1 mark for each extra tick (minimum 0). The other three rows are input validation, authentication and testing, not maintainability.
1(b) J277 2.3.1	1 mark each to max 2: Explain what a section of code does / the purpose of the code Help other programmers (in a team) understand the code (quickly) Help the original programmer understand / remember the code when they return to it later Make errors easier to find and fix / make it easier to update or change the code Explain how a complex part / algorithm works / why it was written that way Can be used to temporarily stop a line of code running when testing (commenting out)	**2**	Two different reasons are needed: "easier to read" and "easier to understand" are the same point. Do not accept "makes the program run faster / better" or "makes the program more secure". Credit any other valid reason.
2(a) J277 2.3.1	1 mark each to max 2: Shows which statements are inside the `for` loop / are repeated 7 times (lines 04 to 08) Shows that line 07 only runs when the `if` condition is true / belongs to the `if` (selection) Makes the structure / where each block starts and ends easy to see, so another programmer can follow the code / find errors (such as a missing `endif` or `next`) more easily	**2**	"Easier to read" alone, with no reference to the loop, the `if` or blocks of code: max 1. Do not accept "the program will not run without it" (true of Python, but not the point; ERL does not need indentation).

QUESTION	ANSWER	MARKS	GUIDANCE

2(b)
J277 2.3.1
	1 mark each to max 5:
	Meaningful identifiers for `t` and `h`, e.g. `totalRain`, `heavyDays`
	Meaningful identifiers for `r` and `d`, e.g. `rainfall`, `day`
	New identifiers follow a naming convention (no spaces, one consistent style such as camelCase) and are used consistently everywhere (every use changed, same spelling)
	Lines 04 to 08 indented inside the `for` loop
	Line 07 indented further than the `if` (inside the `if`)
	At least one relevant comment that explains the purpose of the program or of a section (e.g. what counts as a heavy-rain day)
	`// Adds up a week of rainfall and counts heavy-rain days`
	`totalRain = 0`
	`heavyDays = 0`
	`for day = 1 to 7`
	`    rainfall = float(input("Enter rainfall in mm"))`
	`    totalRain = totalRain + rainfall`
	`    // more than 25 mm in a day is heavy rain`
	`    if rainfall > 25 then`
	`        heavyDays = heavyDays + 1`
	`    endif`
	`next day`
	`print("Total rainfall: " + str(totalRain))`
	`print("Heavy-rain days: " + str(heavyDays))`
		5	The program must still do the same job: if the rewrite changes the logic (e.g. `>= 25`, loop runs a different number of times), max 4. Allow `i` or `count` for the loop counter. Allow snake_case (`total_rain`) or any other consistent convention. Do not accept identifiers with spaces. A comment that only repeats the code (e.g. `// add 1 to h`) does not get the comment mark. Allow # comments. Allow inline comments at the end of a line.

3(a)
J277 2.2.3, 2.3.1
	1 mark each:
	Checks if `cost > 100`
	... and if so sets `cost` to 90% of its value / takes 10% off (`cost * 0.9` / `cost - cost * 0.1`)
	Returns the (new) cost in all cases
	`function applyDiscount(cost)`
	`    if cost > 100 then`
	`        cost = cost * 0.9`
	`    endif`
	`    return cost`
	`endfunction`
		3	Do not accept `>= 100`. Allow a new variable for the result (e.g. `newCost`) if it is set in both cases and returned. Allow `return cost * 0.9` inside the `if` with `return cost` after it. Printing the cost instead of returning it: do not award the third mark. Inputting the cost inside the function (overwriting the parameter): do not award the first mark.

3(b) J277 2.2.3	`adultCost = applyDiscount(adultCost)`	**1**	The returned value must be stored in `adultCost` (or in a new variable that is then used in line 11). Do not accept `applyDiscount(adultCost)` on its own. Do not accept `seniorCost` as the parameter.

QUESTION	ANSWER	MARKS	GUIDANCE
3(c) J277 2.3.1	1 mark each to max 3: The discount code is written once instead of twice / less repeated code / the program is shorter If the discount rule changes (e.g. a different percentage or limit), it only has to be changed in one place ... so there is less chance of changing one copy and missing the other / the two ticket types cannot end up with different rules by mistake The function can be tested on its own / errors in the discount only need fixing in one place The function name describes what the code does, so another programmer can understand line 03 / the program more easily The function can be reused / called again, e.g. if a new ticket type is added, without writing the discount code again	**3**	Answers must be about maintainability. Do not accept "the program runs faster" or "uses less memory". Generic points about sub programs with no link to this program (the discount, the two ticket types): max 2. Credit any other valid point.
4(a) J277 2.3.1	1 mark each to max 2: All the code looks the same / is consistent, so any programmer can read and understand code written by someone else Programmers can tell what kind of item an identifier is (e.g. a variable or a constant) from its style Fewer mistakes from names written in different styles (e.g. `ticketsSold`, `TicketsSold` and `tickets_sold` being treated as different identifiers) Saves time when code is changed / updated / debugged by a different member of the team	**2**	Do not accept "the program will run faster" or "makes the program more secure". "Easier to read" alone: 1 mark only. Credit any other valid point.

Total: 20 marks