Chhetri Academy

chhetriacademy.com

GCSE COMPUTER SCIENCE

2.5

OCR J277 | Specification point 2.5.1

Languages

Untiered

Answers and examiner comments

3	2023-2025	Untiered
question groups	available papers	tier

Using this booklet

1. Answer numbers match the companion question booklet.
2. Follow the original question number on marking tables.
3. Examiner comments are included where available.

Name _____ **Date** _____

Some entries use full source marking material in an appendix. Any unavailable mark schemes are identified beside the relevant answer.

<table>
<tr><th colspan="3">Question</th><th>Answer</th><th>Mark</th><th>Guidance</th></tr>
<tr><td>1</td><td>(a)</td><td></td><td>1 mark per row

<table>
<tr><th>Statement</th><th>Low-level</th><th>High-level</th></tr>
<tr><td>The same language can be used on computers that use different hardware</td><td></td><td>✓</td></tr>
<tr><td>It allows the user to directly manipulate memory</td><td>✓</td><td></td></tr>
<tr><td>It allows the user to write English-like words</td><td></td><td>✓</td></tr>
<tr><td>It always needs to be translated into object code or machine code</td><td></td><td>✓</td></tr>
</table></td><td>4
(AO1 1b)</td><td>No mark if more than 1 tick for that row.

Allow other indications of choice (e.g. cross) as long as clear.</td></tr>
<tr><td>1</td><td>(b)</td><td></td><td><u>total</u> = num1 + num2</td><td>1
(AO3 2b)</td><td>Allow other logically valid responses that result in <code>total</code> storing the correct value. Accept other suitable assignment operators (e.g. ←)

e.g.
<code>total = sum(num1, num2)</code>

<code>total = num2 + num1</code>

<code>x = num1 + num2</code>
<code>total = x</code>

Ignore any values given to the variable. Ignore capitalisation and minor misspelling. Ignore superfluous code that does not affect outcome.</td></tr>
</table>

Question			Answer	Mark	Guidance
1	(c)	(i)	`print(12 ^  2)`	1 (AO2 1a)	Accept ** or other sensible operator that indicates raising to a power. If pseudocode operator given, must be a single word/symbol (e.g. `pow`), not containing spaces.
1	(c)	(ii)	`if number MOD 2 == 0 then`	1 (AO2 1a)	Accept **%** or other sensible operator that indicates modulus If pseudocode operator given, must be a single word/symbol (e.g. `modulo`), not containing spaces.
1	(c)	(iii)	`difference = measurement1 - measurement2`	1 (AO2 1a)	Accept other sensible operator that indicates subtraction. If a pseudocode operator given, must be a single word/symbol (e.g. `minus`), not containing spaces.

<table>
<tr><th colspan="3">Question</th><th>Answer</th><th>Mark</th><th>Guidance</th></tr>
<tr><td>1</td><td>(d)</td><td></td><td>1 mark each:
• Start is set to 3 on line 01 and 3 is output on line 03.
• 2, 1 and 0 are output on next 3 iterations with start updated to 2, 1, 0, -1 on correct line numbers.
• Finished is output on line 06

<table>
<tr><th>Line</th><th>start</th><th>Output</th><th></th></tr>
<tr><td>01</td><td>3</td><td></td><td rowspan="2">BP1</td></tr>
<tr><td>03</td><td></td><td>3</td></tr>
<tr><td>04</td><td>2</td><td></td><td rowspan="8">BP2</td></tr>
<tr><td>03</td><td></td><td>2</td></tr>
<tr><td>04</td><td>1</td><td></td></tr>
<tr><td>03</td><td></td><td>1</td></tr>
<tr><td>04</td><td>0</td><td></td></tr>
<tr><td>03</td><td></td><td>0</td></tr>
<tr><td>04</td><td>-1</td><td></td></tr>
<tr><td>06</td><td></td><td>Finished</td><td>BP3</td></tr>
</table>
</td><td>3
(AO3 2c)</td><td>Ignore lines 02 and 05 in answer unless these change or output any values.

Candidate may repeat start value when unchanged, this is acceptable.

Penalise incorrect or missing line numbers or additional output once only then FT. This includes where variable change and output appear on the same line.

-1 must not be output for BP2

Penalise missing or incorrect output once only for BP1 and FT for missing or incorrect output for BP2.

Finished may be with or without quotes. Ignore case or minor spelling error.

Max 2 marks if any incorrect output or changes to <code>start</code>.

Do not accept calculated values of start (e.g. <code>3-1</code>)</td></tr>
</table>

Original question 7

The full source mark scheme begins on page 9. Use the original paper code, tier and question number shown above.

Question			Answer	Mark	Guidance
2	(a)		• 2 • 6	2	
2	(b)	(i)	2 marks per construct max • FOR • Count controlled • repeats a specified number of times • WHILE • Condition controlled • repeats **while** a condition is **true** / **until** a condition is **false** // condition tested before code runs • DO UNTIL / REPEAT UNTIL • Condition controlled • repeats **until** a condition is **true** / **while** a condition is **false** // condition tested after code runs	4	Marks are independent, allow mark for description even if example/type of loop not given. Allow any 2 per construct. Allow suitable examples from other high-level languages. If WHILE and DO UNTIL given, allow “condition controlled” as description of both. Do not give “condition controlled” twice if no other differentiation between types of iteration. Do not allow “repeats infinitely” – this is only true if the condition is never met. Do not allow mismatched answers (e.g. “a while loop repeats until a condition is true”)
2	(b)	(ii)	• Sequence • Selection	2	Allow slight alternatives (e.g. sequencing) Do not allow examples by themselves (e.g. IF) Allow branching as an alternative to selection.

Question			Answer	Mark	Guidance
2	(c)		1 mark for tool 1 mark for matching description e.g. • Text/code editor… • …allows program code to be **written** / **entered / changed** • …allows errors to be fixed • Debugger / error reporting / diagnostics… • …identifies / finds / checks for errors • …shows location / detail of errors • …suggests fixes • Run-time environment / output window… • …allows **program / code** to be run / executed • …shows output of the **program / code** • Translator / compiler / interpreter … • … convert to **low-level/machine code** • …allow program to be executed / run • …produce executable file (only for compiler) • …stops execution when error found (interpreter only) • Stepping … • … execute/run the program line by line	4	Allow other sensible names for features. Description must add more than is given in the identification of the feature to be awarded. For example, “keyword highlighting, highlights keywords” is 1 mark for the feature only. If compiler and interpreter given as two distinct features, allow both (with suitable descriptions). Do not allow translator and compiler/interpreter. **Description must match tool. If tool name wrong / missing, do not give description.** Allow sensible references to AI where appropriate. Allow other sensible features of an IDE (e.g. line numbering, **auto** indent, collapsed blocks, etc) with suitable description. Description must add (slightly) more than just repeating the feature name. Do not allow features of programming languages (comments, casting, etc). For text editor / error diagnostics / debugger, allow other sensible features listed as features in the mark scheme as description (e.g. “text editor does pretty printing”, “debugger does stepping”)

Question			Answer	Mark	Guidance
			• Variable watch… • … see the **contents/data** held in variables / see how (contents of) variables **change** • Break points … • … will allow the program to stop **at a chosen / set position** • Pretty printing // keyword highlighting… • … allows keywords / variables to be coloured / identified • Keyword completion // autocomplete // autocorrect // syntax suggestion… • …suggests/corrects code/syntax (when first part entered). • GUI (builder)… • …allows creation of user interface / buttons / by example		

PMT

GCSE

Computer Science

J277/02: Computational thinking, algorithms and programming

General Certificate of Secondary Education

Mark Scheme for June 2024

Oxford Cambridge and RSA Examinations

PMT

OCR (Oxford Cambridge and RSA) is a leading UK awarding body, providing a wide range of qualifications to meet the needs of candidates of all ages and abilities. OCR qualifications include AS/A Levels, Diplomas, GCSEs, Cambridge Nationals, Cambridge Technicals, Functional Skills, Key Skills, Entry Level qualifications, NVQs and vocational qualifications in areas such as IT, business, languages, teaching/training, administration and secretarial skills.

It is also responsible for developing new specifications to meet national requirements and the needs of students and teachers. OCR is a not-for-profit organisation; any surplus made is invested back into the establishment to help towards the development of qualifications and support, which keep pace with the changing needs of today's society.

This mark scheme is published as an aid to teachers and students, to indicate the requirements of the examination. It shows the basis on which marks were awarded by examiners. It does not indicate the details of the discussions which took place at an examiners' meeting before marking commenced.

All examiners are instructed that alternative correct answers and unexpected approaches in candidates' scripts must be given marks that fairly reflect the relevant knowledge and skills demonstrated.

Mark schemes should be read in conjunction with the published question papers and the report on the examination.

© OCR 2024

PMT

J277/02 Mark Scheme June 2024

MARKING INSTRUCTIONS

PREPARATION FOR MARKING
RM ASSESSOR

1. Make sure that you have accessed and completed the relevant training packages for on-screen marking: *RM Assessor Assessor Online Training*; *OCR Essential Guide to Marking*.

2. Make sure that you have read and understood the mark scheme and the question paper for this unit. These are posted on the RM Cambridge Assessment Support Portal http://www.rm.com/support/ca

3. Log-in to RM Assessor and mark the **required number** of practice responses ("scripts") and the **number of required** standardisation responses.

   YOU MUST MARK 10 STANDARDISATION RESPONSES BEFORE YOU CAN BE APPROVED TO MARK LIVE SCRIPTS.

MARKING

1. Mark strictly to the mark scheme.

2. Marks awarded must relate directly to the marking criteria.

3. The schedule of dates is very important. It is essential that you meet the RM Assessor 50% and 100% (traditional 40% Batch 1 and 100% Batch 2) deadlines. If you experience problems, you must contact your Team Leader (Supervisor) without delay.

4. If you are in any doubt about applying the mark scheme, consult your Team Leader by telephone or the RM Assessor messaging system, or by email.

5. **Crossed Out Responses**
   Where a candidate has crossed out a response and provided a clear alternative then the crossed out response is not marked. Where no alternative response has been provided, examiners may give candidates the benefit of the doubt and mark the crossed out response where legible.

   Rubric Error Responses – Optional Questions
   Where candidates have a choice of question across a whole paper or a whole section and have provided more answers than required, then all responses are marked and the highest mark allowable within the rubric is given. Enter a mark for each question answered into RM assessor, which will select the highest mark from those awarded. *(The underlying assumption is that the candidate has penalised themselves by attempting more questions than necessary in the time allowed.)*

Multiple Choice Question Responses
When a multiple choice question has only a single, correct response and a candidate provides two responses (even if one of these responses is correct), then no mark should be awarded (as it is not possible to determine which was the first response selected by the candidate).
When a question requires candidates to select more than one option/multiple options, then local marking arrangements need to ensure consistency of approach.

Contradictory Responses
When a candidate provides contradictory responses, then no mark should be awarded, even if one of the answers is correct.

Short Answer Questions (requiring only a list by way of a response, usually worth only **one mark per response**)
Where candidates are required to provide a set number of short answer responses then only the set number of responses should be marked. The response space should be marked from left to right on each line and then line by line until the required number of responses have been considered. The remaining responses should not then be marked. Examiners will have to apply judgement as to whether a 'second response' on a line is a development of the 'first response', rather than a separate, discrete response. *(The underlying assumption is that the candidate is attempting to hedge their bets and therefore getting undue benefit rather than engaging with the question and giving the most relevant/correct responses.)*

Short Answer Questions (requiring a more developed response, worth **two or more marks**)
If the candidates are required to provide a description of, say, three items or factors and four items or factors are provided, then mark on a similar basis – that is downwards (as it is unlikely in this situation that a candidate will provide more than one response in each section of the response space.)

Longer Answer Questions (requiring a developed response)
Where candidates have provided two (or more) responses to a medium or high tariff question which only required a single (developed) response and not crossed out the first response, then only the first response should be marked. Examiners will need to apply professional judgement as to whether the second (or a subsequent) response is a 'new start' or simply a poorly expressed continuation of the first response.

6. Always check the pages (and additional objects if present) at the end of the response in case any answers have been continued there. If the candidate has continued an answer there, then add a tick to confirm that the work has been seen.

7. Award No Response (NR) if:
   - there is no attempt to answer the question (including blank responses and comments such as "I don't know")

   Award Zero '0' if:
   - An attempt is made in the answer space but this is not worthy of credit.

   Team Leaders must confirm the correct use of the NR button with their markers before live marking commences and should check this when reviewing scripts.

8. The RM Assessor **comments box** is used by your team leader to explain the marking of the practice responses. Please refer to these comments when checking your practice responses. **Do not use the comments box for any other reason.**
   If you have any questions or comments for your team leader, use the phone, the RM Assessor messaging system, or e-mail.

9. Assistant Examiners will send a brief report on the performance of candidates to their Team Leader (Supervisor) via email by the end of the marking period. The report should contain notes on particular strengths displayed as well as common errors or weaknesses. Constructive criticism of the question paper/mark scheme is also appreciated.

10. For answers marked by levels of response (not applicable in J277/02):
    a. **To determine the level** – start at the highest level and work down until you reach the level that matches the answer
    b. **To determine the mark within the level**, consider the following

Descriptor	Award mark
On the borderline of this level and the one below	At bottom of level
Just enough achievement on balance for this level	Above bottom and either below middle or at middle of level (depending on number of marks available)
Meets the criteria but with some slight inconsistency	Above middle and either below top of level or at middle of level (depending on number of marks available)
Consistently meets the criteria for this level	At top of level

PMT

J277/02 Mark Scheme June 2024

11. **Annotations**

Annotation	Meaning
A	Omission mark
BOD	Benefit of doubt (must be accompanied with a tick)
	Cross
FT	Follow through (must be accompanied with a tick)
NAQ	Not answered question
NBOD	Benefit of doubt not given
REP	Repeat
	Tick
TV	Too vague
SEEN	Blank pages, pages with no annotation, no attempt to answer the question, page seen on QER

12. **Subject Specific Marking Instructions**

Mark scheme conventions:

- Each mark point is worth 1 mark unless stated otherwise
- Each mark point can only be awarded once
- A word/phrase that is underlined needs to be exact in the answer to award the mark point
- A word/phrase that is **bold** needs that concept to be in the answer (but can be given in multiple ways) to award the mark point
- 3 dots at the end of one mark point and at the start of the next mark point mean that the second mark point cannot be awarded without the first being awarded, unless the mark scheme states otherwise (for example a reasonable attempt with some inaccuracies)
- 3 dots at the start of a mark point, without 3 dots at the end of the mark point above, means the sentence carries on and there is no dependency
- Any text in brackets is not required to gain the mark point
- Single / means alternative word
- Double // means an alternative statement that is acceptable for the same mark point
- Enlarged font is used for visibility reasons only

Annotating scripts:

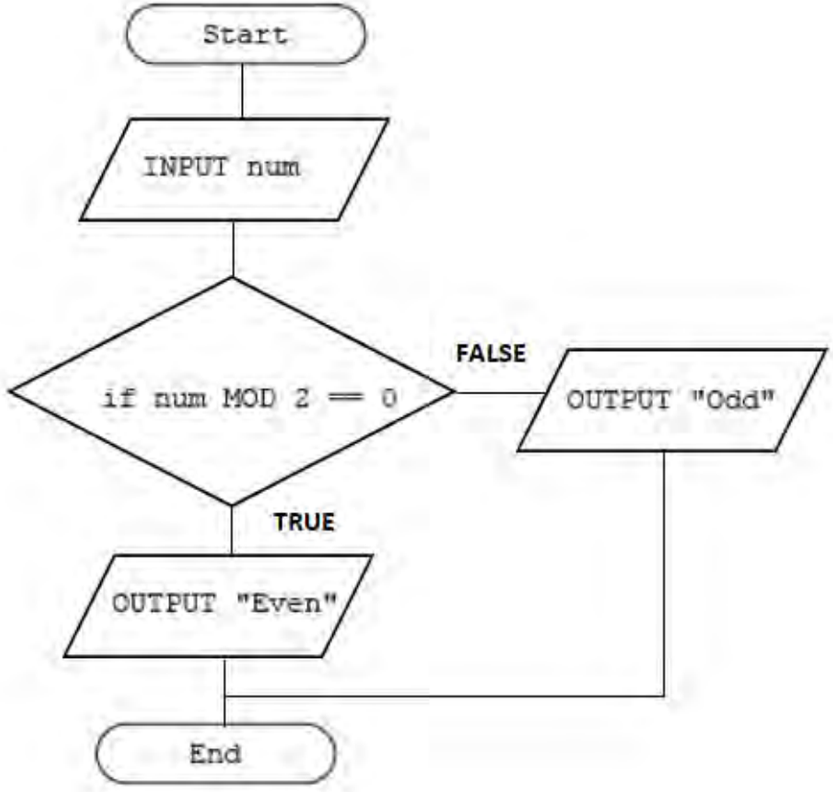
- Blank pages at the start of the script need SEEN annotation
- Any questions answered elsewhere (e.g. on the first blank pages, separately on the page) need to be linked within RM Assessor and annotated with ticks/crosses/SEEN as appropriate
- 1 tick for every mark awarded, if a question is given 3 marks there must be 3 ticks.
- A BOD or FT annotation needs to be accompanied by a tick
- Any answers with no candidate response need a SEEN annotation and NR entered as the mark.
- Any questions where the candidate has not attempted the question e.g. answered 'don't know' need a SEEN annotation and NR entered as the mark.
- All questions must be annotated throughout the marking process.

PMT

J277/02 Mark Scheme June 2024

Question			Answer	Mark	Guidance
1			<table><tr><th rowspan="2">Keyword</th><th colspan="2">Programming construct</th></tr><tr><th>selection</th><th>iteration</th></tr><tr><td>`if`</td><td>✓</td><td></td></tr><tr><td>`for`</td><td></td><td>✓</td></tr><tr><td>`while`</td><td></td><td>✓</td></tr></table>	3 (AO1)	

J277/02 Mark Scheme June 2024

2			• Correct shape for **all three** inputs AND outputs (parallelogram) • Correct shape for decision (diamond) • True and False // Yes and No labelled **correctly** *(true/Yes linking to "Even")* • **All** lines joined up correctly and link to End. 	4 (AO2)	No need for arrows – lines are acceptable. BOD for correct answers that include a loop back to the start

J277/02 Mark Scheme June 2024

3	(a)		Max 1 mark for definition that is clearly different from a logic error. • (an error that) breaks the **rules/grammar** of the **programming language** • Stops the program from **running** // does not allow program to **run** // crashes the program // does not allow program to **translate** Suitable example for 1 mark, e.g. • misspelling **key word** (e.g. **`printt`** instead of **`print`**) • Missing / extra symbol (e.g. missing bracket, missing semicolon) • Mismatched quotes • Invalid variable or function names (e.g. variable starting with a number or including a space) • Incorrect use of operators • Use of reserved keywords for variables (e.g. **`print = 3`**) • Incorrect capitalisation of keywords (e.g. **`P`**`rint` instead of `print`) • Incorrect indentation (of code blocks) • Missing concatenation (e.g. `print(score x)`)	2 (AO1)	BOD code/program etc for BP1 Do not allow answers linked to data types. "incorrect grammar" by itself is NE Do not allow “stop working”, "does not work", etc – TV. Do not accept missing quotation marks e.g. `print(hello)` (could be a variable name) BOD given code that could cause a syntax error in a high-level language.

J277/02 Mark Scheme June 2024

3	**(b)**		1 mark each
			• line 03
			• `total = num1 + num2`
			
			• Line 04
			• `if total >= 10 and total <=20 then`
			
			Allow other logical equivalent code e.g.
			`total = int(num1) + int(num2)`
			`if 10 <= total <= 20`
				4 (AO3)	Allow other logical corrections that will fix the problem identified and does not introduce any further errors.
					
					Allow descriptions of changes as long as clear exactly what will change. Do not allow ambiguous descriptions of changes to code.
					
					Ignore missing `then` from line 04.
					
					Ignore capitalisation.

3	**(c)**	**(i)**	1 mark each • Compare to / pick out middle value (which is 6) • **discard only left side // retain only right side** (because 6 < 10)… • …Compare to / pick out (middle value which is) **10**	3 (AO2)	BP1 can be given for generic answer. BP2 and 3 must be linked to data set given For BP2, must remove 1, 2,5 and 6 from list if discussing individual numbers. Allow FT for BP3 if this done incorrectly.

		(ii)	• Data must be sorted / in order	1 (AO1)	
		(iii)	• Merge sort	1 (AO1)	
4	**(a)**		Input e.g. • **Name / keyword** for video (to be searched for) // search **text** • **Controls** for watching video (e.g. play / pause) • **Rating** given to video Output e.g. • Video to be watched // audio • **Results** of search • (total / overall / average) rating of video • Number of views (of video) • Confirmation of data entry / data validity • Messages to user // example messages (e.g "enter a rating", "your rating has been saved") in **quotation marks**	2 (AO1)	1 mark for a suitable input, 1 mark for a suitable output Allow input / print pseudocode statements if meets mark point(s). Does not have to be valid pseudocode. Do not allow examples of inputs (e.g. "music videos")

J277/02 Mark Scheme June 2024

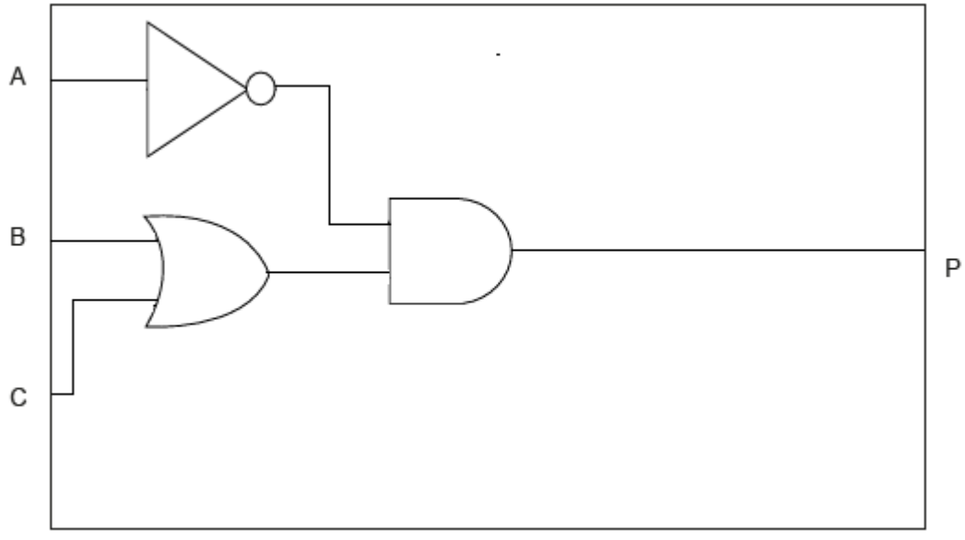
4	**(b)**		Only 1 method asked for. Could be name and description/example or description and example • Authentication • …checking users allowed to access the site / know identity of users • … by example (e.g. username and password) • Anticipating misuse // preventing misuse • ….stopping the user breaking / hacking into the system • …by example (e.g. restricting entry to integers) • Validation • …check / only allow sensible data to be entered / check data is sensible • …by example (e.g. restrict ratings to 1 to 10 / presence check / format check) • Input sanitisation • …removing invalid/special characters • …by example (e.g. remove quotation marks / semicolons) • Maintainability • ...ensuring program is able to be understood by others • ...by example (e.g. modularisation / comments)	2 (AO1)	Allow validation / input sanitisation / passwords as expansion of anticipating misuse. Allow mark for description with no / incorrect name Allow any 2 points from mark scheme as long as clearly linked to a single defensive design method.

J277/02 Mark Scheme June 2024

5	**(a)**		1 mark per group of 2 rows	4 (AO2)	Accept True / False etc.

A	**B**	**C**	**P**	
0	0	0	**0**	1 mark
0	0	1	**1**	
0	1	0	**0**	1 mark
0	1	1	**1**	
1	0	0	**0**	1 mark
1	0	1	**1**	
1	1	0	**1**	1 mark
1	1	1	**1**	

J277/02 Mark Scheme June 2024

5	**(b)**		1 mark each • NOT A • B OR C • AND gate with **two inputs**  	3 (AO3)	**Max 2** if not logically correct or any additional / missing gates. Shapes of gates must be correct with correct number of inputs. Ignore annotation of gate names. NOT gate must include circle. Other gates must not include circle.

PMT

J277/02 Mark Scheme June 2024

<table>
<tr><td>6</td><td>(a)</td><td></td><td>1 mark for each output

<table><tr><td>print(message.upper)</td><td>ABCD1234 (upper case)</td></tr><tr><td>print(message.left(4))</td><td>abcd (lower case)</td></tr><tr><td>print(int(message.right(4))*2)</td><td>2468</td></tr></table></td><td>3
(AO2)</td><td>Case must be correct but BOD if ambiguous.

Allow quotation marks in answer.</td></tr>
<tr><td>6</td><td>(b)</td><td></td><td>1 mark per bullet point :
• storing both strings correctly in word1 and word2
• correct concatenation (word1 then word2)...
• …storing in variable message

Example
word1 = "Hello"
word2 = "Everyone"
message = word1 + word2</td><td>3
(AO3)</td><td>Accept & / + / . etc as valid methods of concatenation. Allow use of sensible concatenation functions e.g. concat() . Do not allow commas.

Do not allow == for assigning value to string. Do not allow spaces in variable names. Penalise once then FT.

Ignore additional code given. Ignore case.

Reasonable attempt at BP2 needed to access BP3.</td></tr>
</table>

7	(a)		1 mark each to max 2 • (machine code) does not need to be translated / compiled / interpreted • Direct control of **hardware / memory** • Faster execution time • Code can be optimised / shorter code / use less memory • Can program for specific hardware • **Assembly language** is fast to translate.	2 (AO1)	"More efficient" by itself is TV. Mark first answer on each line. BP6 relates to Assembly language being a one-to-one direct mapping to machine code.
7	(b)		1 mark each to max 3 • Can produce an **executable file** • program/code **runs/executes** faster (than interpreted version) • end users do not need translator • Can be run **again/multiple times** without re-translating // only needs to translate **once** • **End users** have no access to **source code** // distributed **with no source code**... • …cannot steal/copy/modify code/program • Shows **all/multiple** errors // shows errors **at the end** (of compilation) // creates **error file** • Compiler can optimise the code	3 (AO1)	Allow in reverse (e.g. “interpreter translates every time”) Do not allow "no access to source code" unless clearly talking about end user. Allow if in context of distribution. Do not allow descriptions of how a compiler translates (e.g. "translates whole code in one go") “Faster / quicker” by itself is TV

J277/02 Mark Scheme June 2024

8	(a)		2 marks max per group • Meaningful **identifiers** // meaningful variable names • ...to describe/show what they **store // purpose** of variable • An example of a meaningful variable identifier for this algorithm • Comments • ...to make it easier for other programmers to **follow** / **understand** (part of) the code // explains what the code does // easier to **debug** • An example of a suitable comment for this algorithm • Use of subroutines • ...to **reuse** blocks of code // make code easier to follow • An example of a subroutine for this algorithm • Use of constants • ...to store data that will not change (during program execution) // so data can be changed in one place only • An example of a constant for this algorithm (e.g. store 512 as a constant)	4 (AO2)	Do not accept "what variables do" – incorrect verb, variables store/hold data. BOD notes (and alternatives) for comments. Do not allow instructions. Do not allow indentation (already done in program given) Allow whitespace / **blank** lines (same expansions as comments) Do not award expansion without being clear which method is being discussed. "Makes it easier to understand" by itself is TV.

J277/02 Mark Scheme June 2024

8	(b)		1 mark each to max 6
			• Appropriate use of **both parameters** and **no additional inputs** / **incorrect overwrites** that affect outcome of algorithm
			• Attempt at selection…
			• …correctly checking if `direction` is "left" **and** subtracting 5 from `position` (or equivalent)
			• …correctly checking if `direction` is "right" **and** adding 5 to `position` (or equivalent)
			• Ensuring position (or equivalent) is between 1 and 512 inclusive
			• **Returning** the updated position
			
			Example
			`if direction == "left" then`
			`  position = position - 5`
			`elseif direction == "right" then`
			`  position = position + 5`
			`endif`
			
			`if position < 1 then`
			`  position = 1`
			`elseif position > 512 then`
			`  position = 512`
			`endif`
			
			`return position`
				6 (AO3)	Allow `else` for BP3/4 (validated in question 8a)
					
					Allow <=, >= and equivalents (e.g. <= 0) for BP5.
					
					Do not award BP5 if before BP3 and 4 (otherwise will alter position value)
					
					BP6 only to be given if attempt made at calculating new position. Calculation can be partial/incorrect.
					
					Ignore repeat of function header / end.
					
					Accept flowchart / structured English but must not just repeat the question.
					
					If response uses loop to incorrectly change position multiple times, do not award BP1 (incorrect overwrite)
					
					For minor syntax errors (e.g. missing quotation marks or == for assignment, spaces in variable names) penalise once then FT.

J277/02 Mark Scheme June 2024

Section B

Question			Answer	Mark	Guidance
9	(a)	(i)	• String • Integer • Real / Float	3 (AO3)	Accept alternative equivalent correct data types (e.g. single/double/decimal for BP3) Do not accept char for BP1
		(ii)	• `theTeam.length() - 1 // 5` • `count` • `studentName` • `True`	4 (AO3)	Accept 6 // `theTeam.length()` for BP1 (Python). Accept alternative length functions e.g. len() Accept **`count = 5`** (and equivalents) for BP1. Accept "True" for BP4. Do not allow obvious spaces in variable names. Ignore capitalisation.

J277/02 Mark Scheme June 2024

<table>
<tr><th colspan="3">Question</th><th>Answer</th><th>Mark</th><th>Guidance</th></tr>
<tr><td></td><td>(b)</td><td></td><td>

<code>javelinThrow</code> set to 14.3 on line 01 and <code>yearGroup</code> set to 10 on line 02
<code>score</code> set to 2 on line 06
<code>score</code> set to 4 on line 11
"The score is 4" output on line 13 with no additional outputs (allow input statements)

Example
<table>
<tr><th>Line number</th><th>javelinThrow</th><th>yearGroup</th><th>score</th><th>Output</th></tr>
<tr><td>01</td><td>14.3</td><td></td><td></td><td></td></tr>
<tr><td>02</td><td></td><td>10</td><td></td><td></td></tr>
<tr><td>06</td><td></td><td></td><td>2</td><td></td></tr>
<tr><td>11</td><td></td><td></td><td>4</td><td></td></tr>
<tr><td>13</td><td></td><td></td><td></td><td>The score is 4</td></tr>
</table>
Answer may include lines where no changes or output happens (i.e. lines 3, 4, 5, 7, 8, 9, 10, 12).

Where variable doesn't change, current value may be repeated on subsequent lines.
</td><td>4 (AO3)</td><td>Max 3 if in wrong order or additional (incorrect) changes. Penalise line numbers once then FT.

Allow FT for BP4 for current value of <code>score</code>.

BP4 must not include comma. Ignore superfluous spaces. Ignore quotation marks.

Treat any entry in output column as an output, even if "x", "-" or "0".</td></tr>
</table>

PMT

J277/02 Mark Scheme June 2024

Question			Answer	Mark	Guidance

	(c)	(i)	• inputs a value from the user and stores/uses
			• checks min value (>= 40.0 // < 40)
			• checks max value (<=180.0 // > 180)
			• …outputs **both** valid / not valid correctly based on checks
			
			Example 1 (checking for valid input)
			`h = input("Enter height jumped")`
			`if h >= 40 and h <= 180 then`
			`  print("valid")`
			`else`
			`  print("not valid")`
			`endif`
			
			Example 2 (checking for invalid input)
			`h = input("Enter height jumped")`
			`if h < 40 or h > 180 then`
			`  print("not valid")`
			`else`
			`  print("valid")`
			`endif`
				4 (AO3)	Answers using `AND/OR` for BP2 and BP3 must be logically correct e.g. `if height >=40 and height <=180`. Do not accept `if height >=40 and <=180`
					
					Answers using OR will reverse output for BP4 (see examples).
					
					BP4 needs reasonable attempt at **either** BP2 or BP3. Need to be sure what is being checked to be able to decide which way around valid/invalid should be.
					
					Allow FT for BP4 if reasonable attempt at validating (must include at least one boundary)
					
					Ignore conversion to int on input. `input` cannot be used as a variable name.
					
					Greater than / less than symbols must be appropriate for a high-level language / ERL. Do not accept => (wrong way around) or ≥ (not available on keyboard). No obvious spaces in variable names. Penalise once and then FT.

PMT

J277/02 Mark Scheme June 2024

Question			Answer	Mark	Guidance
	(c)	(ii)	• Any normal value (between 40 and 180 inclusive) • 40.0 // 180.0 • Any value less then 40 // any value greater that 180 // any non-numeric value	3 (AO3)	No need to include decimals, e.g. accept 50. Ignore cm if given. Answer must be actual data (e.g. 50) and **not** description of data (e.g. "a value between 40 and 180"). If descriptions given, do not accept this as non-numeric for BP3
	(d)		• `TeamName` only in first space • `TblResult` in second space • `WHERE` • `...YearGroup = 11`	4 (AO3)	Max 3 if not in correct order / includes other logical errors. Ignore capitals. Do not accept * or additional fields for BP1 Spelling must be accurate (e.g. not TblResult**s**). No spaces in field names, penalise obvious spaces once and then FT. Allow quotation marks around field names, table name and 11 Accept == for BP4 (invalid SQL but works in some environments)

J277/02 Mark Scheme June 2024

Question			Answer	Mark	Guidance
	(e)	(i)	• any **example** of simplification/removing data or focussing on data (in the design) Examples : - "focus on student names and events" - "ignore data such as students' favourite subjects" - "store year groups instead of ages or DOB" - "shows student IDs instead of full student details"	1 (AO3)	Must be applicable to this program (in the context of students and a sports day), **not** a generic description of what abstraction is. Give BOD where this is unclear.
		(ii)	• any **example** of breaking down the **program** into sections/subroutines • any **example** of breaking down the **database** into tables Examples : - "splits the program up into different events" - "separates the validation routines into subroutines" - "breaks the database down into a table per event"	1 (AO3)	Must be applicable to this program, not a generic description of what decomposition is. Give BOD where this is unclear. Do not give answers discussing splitting into **fields** (e.g. split into StudentID, YearGroup, etc). BOD if answer discusses one table but suggests other tables could be used. Do not give answers relating simply to data being split into smaller groups unless this clearly relates to how data is decomposed into **tables** in the DB. Allow reference to sports day to mean sports day program.

PMT

J277/02 Mark Scheme June 2024

Question			Answer	Mark	Guidance

	(f)		• Input team name AND score **and** store / use separately
			• Attempt at using iteration…
			• …to enter team/score until "stop" entered
			• Calculates highest score
			• Calculates winning team name…
			• …Outputs highest score **and** team name
			
			<u>Example 1</u>
			`highscore = 0`
			`while team != "stop"`
			`  team = input("enter team name")`
			`  score = input("enter score")`
			`  if score > highscore then`
			`  highscore = score`
			`  highteam = team`
			`  endif`
			`endwhile`
			`print(highscore)`
			`print(highteam)`
			
			<u>Example 2 (alternative)</u>
			`scores = []`
			`teams = []`
			`while team != "stop"`
			`  team = input("enter team name")`
			`  score = input("enter score")`
			`  scores.append(score)`
			`  teams.append(team)`
			`endwhile`
			`highscore = max[scores]`
			`highteam = teams[scores.index(highscore)]`
				6 (AO3)	For BP3, allow "stop" to be entered for either team name or score (or both). Allow third input (e.g. "do you wish to stop?")
					
					Allow use of `break` (or equivalent) to exit loop for BP3.
					
					Allow use of recursive function(s) for BP2/3.
					
					Initialisation of variables not needed - assume variables are 0 or empty string if not set.
					
					Ignore that multiple teams could get the same high score, assume only one team has the highest score.
					
					BP4/5 could be done in many ways – see examples. Allow any logically valid method. Allow use of max/sum functions and use of arrays/lists.
					
					FT for BP6 if attempt made at calculating highest score/name
					
					If answer simply asks for multiple entries (not using iteration), BP2 and 3 cannot be accessed but all others available.

PMT

J277/02 Mark Scheme June 2024

Question			Answer	Mark	Guidance
			`print(highscore)` `print(highteam)`		For minor syntax errors (e.g. missing quotation marks or == for assignment, spaces in variable names) penalise once then FT. `input` cannot be used as a variable name.

PMT

Need to get in touch?

If you ever have any questions about OCR qualifications or services (including administration, logistics and teaching) please feel free to get in touch with our customer support centre.

Call us on

01223 553998

Alternatively, you can email us on

support@ocr.org.uk

For more information visit

ocr.org.uk/qualifications/resource-finder
ocr.org.uk
Twitter/ocrexams
/ocrexams
/company/ocr
/ocrexams

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT

OCR is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored. © OCR 2024 Oxford Cambridge and RSA Examinations is a Company Limited by Guarantee. Registered in England. Registered office The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA.

Registered company number 3484466. OCR is an exempt charity.

OCR operates academic and vocational qualifications regulated by Ofqual, Qualifications Wales and CCEA as listed in their qualifications registers including A Levels, GCSEs, Cambridge Technicals and Cambridge Nationals.

OCR provides resources to help you deliver our qualifications. These resources do not represent any particular teaching method we expect you to use. We update our resources regularly and aim to make sure content is accurate but please check the OCR website so that you have the most up-to-date version. OCR cannot be held responsible for any errors or omissions in these resources.

Though we make every effort to check our resources, there may be contradictions between published support and the specification, so it is important that you always use information in the latest specification. We indicate any specification changes within the document itself, change the version number and provide a summary of the changes. If you do notice a discrepancy between the specification and a resource, please contact us.

Whether you already offer OCR qualifications, are new to OCR or are thinking about switching, you can request more information using our Expression of Interest form.

Please get in touch if you want to discuss the accessibility of resources we offer to support you in delivering our qualifications.

Question 1 (a)

1 (a) The table contains four statements about programming languages.

Tick (✓) **one** box in each row to identify whether each statement describes a low-level programming language or a high-level programming language.

Statement	Low-level	High-level
The same language can be used on computers that use different hardware		
It allows the user to directly manipulate memory		
It allows the user to write English-like words		
It always needs to be translated into object code or machine code		

[4]

This question was answered well by many candidates. The strongest responses showed a good understanding of the difference between low-level and high-level languages. Incorrect responses tended to be on the first row related to portability. A high-level language such as Python is portable, with translators available for many different types of processors. A low-level language is specific to one type of processor.

Question 1 (b)

(b) The variables `num1` and `num2` store integers.

Write pseudocode to add the integers stored in `num1` and `num2`. Store the result in a variable with the identifier `total`

.....

..... **[1]**

Candidates appear to be getting more confident at answering simple programming/pseudocode questions such as this. The majority of responses included code written to produce the required outcome. The use of multiple steps was allowed.

Question 1 (d)

(d) Read the following pseudocode algorithm:

```
01  start = 3
02  do
03    print(start)
04    start = start - 1
05  until start == -1
06  print("Finished")
```

Complete the following trace table for the given algorithm.

Line number	`start`	Output

[3]

This question assessed candidates' ability to trace through and understand the steps taken by an algorithm. This also tested their understanding of condition-controlled loops. Many responses were very successful with this and achieved full marks.

Mistakes tended to be with identifying the line number where each change occurred or outputting values that were not actually output (e.g. -1).

Examiners were instructed to only penalise a misunderstanding once. Where (for example) line numbers were incorrect, this would still have allowed 2 out of 3 marks to be achieved.

Question 7 (a) and (b)

7 Programs can be written in high-level languages or low-level languages.

(a) Give **two** reasons why some programs are written in a low-level language.

1 .....

.....

2 .....

.....

[2]

(b) Describe the benefits of using a compiler instead of an interpreter when writing a program.

.....

.....

.....

.....

.....

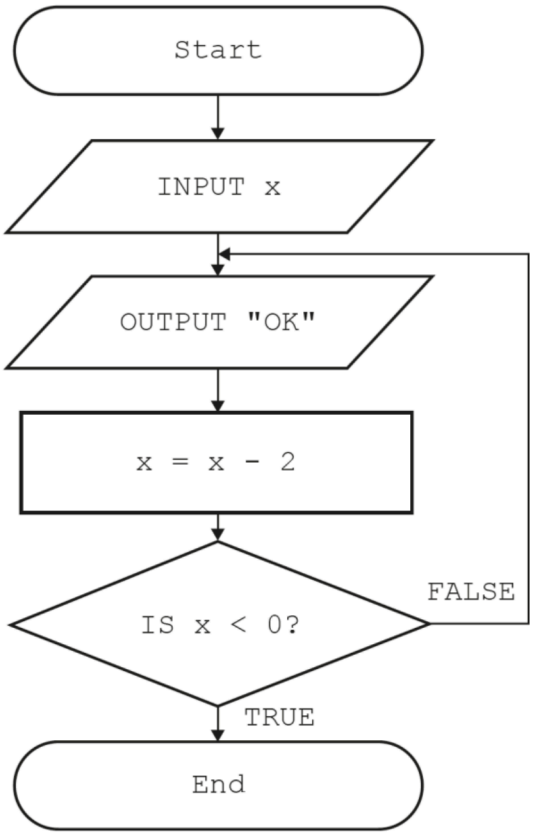
..... **[3]**

These questions assessed understanding of low-level languages and the use of compilers to translate from high-level code. Many candidates were able to do this well and clearly understood the need for high-level languages to be translated into machine code before the processor can execute this. The questions, however, did not ask just for this but specifically asked for reasons and benefits. Where candidates left their response as simply a discussion of the difference between high-level and low-level code or between translators and compilers, it was difficult for examiners to award marks.

Better responses were able to give clear reasons for the use of low-level code (such as direct control of hardware and faster execution of code) and the benefits of using a compiler instead of an interpreter (such as being able to distribute an executable file with no access to source code and not needing to translate code again once this has been done).

Question 2 (a)

2 This flowchart uses iteration.



(a) Complete the table by identifying how many times "OK" will be output for when each value is input.

Input value	Number of times "OK" is output
3	
10	

[2]

This question involved following a flowchart to determine the outcome produced. Most students did this successfully for the input value of 3, determining correctly that OK would be output twice.

However, fewer candidates were able to do this for the input value of 10. This may be due to confusion over the use of the 'less than' operator when the value held in variable **x** was set to zero.

Question 2 (b) (i)

(b) Iteration is a programming construct.

(i) Identify **and** describe **two** different examples of iteration that can be used in a high-level language.

1 .....

.....

.....

2 .....

.....

.....

[4]

Iteration is commonly referred to as loops, or repetition. Candidates were confidently able to name examples of keywords that are used in a high-level language (such as **`for`**, **`while`** or **`do until`**) and describe the use of these well. Many candidates used subject-specific terminology such as **condition controlled** loops or **count controlled** loops when doing this, as well as describing these in more depth.

Some candidates lacked precision with their description of a **`while`** loop, stating that this repeats **until** a condition is **`True`** which is incorrect.

Misconception

A **`while`** loop is condition controlled. It has a condition that is evaluated before each iteration begins; if the condition evaluates to be **`True`**, the loop repeats one more time before the condition is evaluated again.

For example, **`while x > 7`** : The condition here that is evaluated is whether the contents of the variable `x` is larger than 7. As long as this is the case, the loop will continue to repeat.

Many candidates suggested that the while loop should repeat **until** the condition is **`True`**, that is, **`while x > 7`** should repeat for values of **`x`** that are not larger than 7 and only stop repeating when **`x`** is larger than 7. This is clearly incorrect.

Centres and candidates must be aware that their answers should be precise and technically correct if they are to achieve highly.

Question 2 (b) (ii)

(ii) Give the names of **two** other programming constructs that are used in a high-level language.

1 .....

2 .....

[2]

Many candidates were able to list sequence and selection as two other programming constructs which are listed in the specification document. We accepted sensible alternatives (such as branching).

Question 2 (c)

(c) A programmer implements the flowchart as a program in a high-level language using an Integrated Development Environment (IDE).

Identify and describe **two** tools that the programmer could use in the IDE to create a program from the flowchart.

Tool 1 .....

Description .....

.....

Tool 2 .....

Description .....

.....

[4]

Candidates were mostly able to name and describe suitable tools contained within an Integrated Development Environment, such as an editor, error diagnostics, run-time environment or translator.

Many other sensible tools and sub-tools such as a variable watch window, stepping, keyword completion or Artificial Intelligence (AI) tools were all also credited.

A small number of candidates mistakenly described tools contained within programming languages, such as commenting or use of variables. Descriptions were generously given where they added to the tool name, but repetition (such as “a code editor / edits code”) was not given both mark parts.