Chhetri Academy

chhetriacademy.com

GCSE COMPUTER SCIENCE

2.1

OCR J277 | Specification point 2.1.3

Searching and sorting algorithms

Untiered

Answers and examiner comments

10	2018-2025	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.

Question			Answer	Mark	Guidance

4	(a)	(i)	1 mark per filled gap, max 3
			
			`01 function librarycode(title, year)`
			
			`02  parta = title.substring(0, 3)`
			
			`03  partb = year.substring(2, 2)`
			
			`04  return parta.upper + partb`
			
			`05 endfunction`
				3	Ignore capitalisation.
					
					Allow **librarycode =** for 3rd mark – this is an equivalent in some languages for returning a value (eg. Visual Basic).

Question			Answer	Mark	Guidance

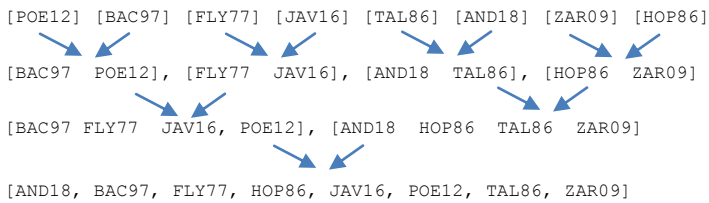
4	(a)	(ii)	1 mark per bullet, max 6
			
			• Input title and year from user
			• Open bookcodes.txt
			• Call the librarycode() function…
			• … with the two parameters that match input values
			• … write out **code obtained** to the text file
			• Close text file
				6	Example algorithm
					
					`title = input("enter title")`
					`year = input ("enter year")`
					`code = librarycode(title, year)`
					`myFile = openWrite("bookcodes.txt")`
					`myFile.writeLine(code)`
					`MyFile.close()`
					
					Note, pseudocode shown above is an example – candidates may answer very differently, but award marks if intention can be seen.
					
					Bullet points 3,4 and 5 could be done in one line:
					`myFile.writeLine(librarycode(title, year))`
					
					Do not award bullet point 3 if candidate is defining the function rather than calling it.
					
					Allow bullet point 2 (opening text file) if correctly referred to during write operation.
					
					Bullet point 3 must include brackets () to signify it is the function being called or indication that is being called.

4	(b)	(i)	1 mark per bullet, max 2. • Function **returns** a value • Procedure **does not return** a value	2	Allow "does not" for second mark if intention is clear (ie if it is obvious that the "not" refers to not returning a value). Allow discussion of how returned value in a function can be used (e.g. to assign to a variable or to use this returned value in some way).

J276/02 Mark Scheme June 2018

Question			Answer	Mark	Guidance
4	(b)	(ii)	1 mark per bullet, max 4. Mark in pairs. e.g. • Breaks down / decomposes / modularises the problem / program // structures the program • …making it easier to design/create/test • …each subroutine can be tested separately • Reuse code (in different programs) • …quicker to develop (new) programs • …build on existing work / use of a library of subroutines • Avoid repetition of code (in the same program) • …makes program shorter / smaller • … subprogram called instead of copying/pasting. • … quicker to develop (new) programs • Easier to maintain …as code is easier to understand/read • …as code is shorter • Easier to debug • …as code is shorter • …same bugs will not have been copied to other areas of the program. • Work can be split up in a team • ...to suit developers' skill set • …to work on different subprogram at the same time / develop separately • Allows for abstraction / removes complexity • …subprograms can be used by programmers who do not need to understand how they work.	4	Maximum of two benefits with expanstions to be marked as per question. Allow other sensible expansions. Allow expansions which cross over to other benefits (e.g. breaks down the problem / to make it easier to maintain). Allow "can be called multiple times" Allow "file size is smaller". Do not allow "more efficient" without further explanation.

J276/02 Mark Scheme June 2018

Question			Answer	Mark	Guidance
4	(c)	(i)	1 mark per bullet, max 4. • List split into individual elements (may be done over several steps or just as a starting point) • **Merge** individual elements into **sorted** lists of size 2 • **Merge** lists of size 2 into **sorted** lists of size 4 • **Merge** lists of size 4 into final **sorted** list.	4	Candidates **can** describe how the merge sort would work rather than showing output values at each stage. Ignore intermediate steps. **Do not** give final mark for simply showing the list sorted. Must have the (correct) idea of where it being merged from previous lists. Candidates' answers describing / showing other sorting algorithms (e.g. bubble sort, insertion sort) are worth 0 marks. 
4	(c)	(ii)	1 mark per bullet, max 2. • Faster/quicker (to sort)… • …for large lists // for lists that are **more unordered** • Has a consistent running time (for a lists of same length)… • …doesn't depend on how ordered original list is	2	Accept (correct) reference to big O notation for 2nd mark on either mark point although this is beyond scope of GCSE specification. Allow "more efficient" for BOD on first bullet point.

Original question 2

The full source mark scheme begins on page 28. Use the original paper code, tier and question number shown above.

Original question 6

The full source mark scheme begins on page 28. Use the original paper code, tier and question number shown above.

<table>
<tr><th colspan="3">Question</th><th>Answer</th><th>Mark</th><th>Guidance</th></tr>
<tr><td>2</td><td>(a)</td><td></td><td>
<table>
<tr><th>Statement</th><th>True (✓)</th><th>False (✓)</th></tr>
<tr><td>The list of words is initially split into a sorted set and an unsorted set</td><td>✓</td><td></td></tr>
<tr><td>The insertion sort uses a divide stage and then a conquer stage.</td><td></td><td>✓</td></tr>
<tr><td>The list of words must be in order before the insertion sort can start</td><td></td><td>✓</td></tr>
<tr><td>Each word is inserted into the correct place in the array, one by one</td><td>✓</td><td></td></tr>
<tr><td>The insertion sort will not work because the word “wall” appears twice.</td><td></td><td>✓</td></tr>
</table>
</td><td>1</td><td>1 mark per row</td></tr>
</table>

	(b)		• Pick middle value / pumpkin // find midpoint • Compare this to house, **no match** • pumpkin>house… • …so discard top half of list // focus on bottom half • Pick **middle value** again, either house or flour… • …finds value // repeat to find value	4	Do not award generic responses except for BP1 Must clearly show the steps taken for this list to achieve more than 1 mark. Do not award "splits the list in half" for BP1 or 4 – incorrect Allow diagrams to demonstrate the process Allow reasonable attempt at BP3 to allow access to BP4

<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 correct row

<table>
<tr><th>OCR Reference Language code</th><th>Selection</th><th>Iteration</th></tr>
<tr><td><pre>for i = 1 to 10
    print(i)
next i</pre></td><td></td><td>✓</td></tr>
<tr><td><pre>whilescore != 0
    playgame()
endwhile</pre></td><td></td><td>✓</td></tr>
<tr><td><pre>if playerHit() then
    score = score - 1
endif</pre></td><td>✓</td><td></td></tr>
<tr><td><pre>switch bonus:
    case 0:
        score = 9
    case 1:
        score = 7
    case 2:
        score = 5
endswitch</pre></td><td>✓</td><td></td></tr>
</table></td><td>4
(AO2 1b)</td><td>No mark given if both boxes in a row ticked.

Accept any response (ticks, crosses, etc) that clearly indicates candidate’s choice.</td></tr>
<tr><td>1</td><td>(b)</td><td></td><td>• <code>score = score + 1 // score +=1 // score++</code></td><td>1
(AO3 2b)</td><td>Allow other logically correct answers that result in score increasing by one and being overwritten. Do not accept <code>score + 1 / score = +1</code>

Accept valid structured English answers that refer to score increasing and overwriting the existing value by one. e.g. “score becomes/equals score plus one”

Ignore any superfluous code that does not affect the outcon</td></tr>
<tr><td></td><td>(c)</td><td></td><td>• Decomposition</td><td></td><td>Correct answer only. Ignore spelling errors.</td></tr>
</table>

			• Abstraction	**(AO1 1a)**	

Question			Answer	Mark	Guidance
3	(a)		• Merge into correct **sorted** lists of **size 2** (12 45 / -99 100 / -13 0 / -27 17) • Merge into correct **sorted** lists of **size 4** (-99 12 45 100 / -27 -13 0 17) … • …Merge into correct **sorted** list of **size 8** (-99 -27 -13 0 12 17 45 100)	3 **(AO2 1b)**	Do not credit BP3 simply for a sorted list. Groups of numbers must clearly be the correct size. Do not all allow answers that show lists being merged and then sorting in place, this is incorrect.
3	(b)		Any four bullet points for 1 mark each • Select / choose / pick **middle** number (or left/right of middle as even number) and … • **…**check if selected number is **equal to / matches** target number *(not just compare)* • …if searched number is **larger**, discard left half // if searched number is **smaller**, discard right half • Repeat until **number found** • … or remaining list is of size 1 / 0 (number not found)	4 **(AO1 1b)**	Do not allow “split the list in half” on its own as first step this is incorrect. Can get BP1 and 2 in one step (e.g. “check if the middle number is the one we’re looking for”) For BP3, accept focussing on correct half Repeat (BP4) must be in the context of an attempt at a binary search. Allow correct reference to recursion. “until number is not in the list” is NE for final BP. Need to explain how this is known.
3	(c)		1 mark each • Starting with the **first value** • Checking **all values in order**	2 **(AO1 1b)**	2nd bullet point must cover both ideas of checking all of the values AND being done in order. “Checks each value” / “one by one” / “step by step” by itself is NE, does not say in order. Do not accept “repeat until value found” for BP2 (question says number is not in the list) “Checks each value from beginning to end” implies order so gets both BP1 and BP2.

Question			Answer	Mark	Guidance
3	(a)		1 mark each • stores/holds **data/value/name/names[pos]** • …so (value) can be changed / swapped / moved / overwritten / inserted • …without being lost. • will be assigned to `names[pos-1]`	**2** (AO2 1b)	Do not allow answers that clearly refer storing the position / index (or any other out of context data) for BP1; it is the name itself that is being stored, not the position. If unclear, allow BOD. e.g. do not allow "holds the values of the index / holds value for position of the name". Allow FT for subsequent points.
3	(b)		1 mark • **do not know** how many iterations / swaps needed • **do not know** (at run time) how many times the value will **change positions** • **do not know** how many times a condition-controlled loop will need to **run** / **execute** 1 mark • condition controlled loops run while/until a condition **is true** / **is false** / until a condition **is met** • repeats while value in [pos-1] is larger than value in [pos] // while (further) swap needed • will swap value until in **correct position** // will swap whilst in **incorrect position**	**2** (AO2 1b	Max 1 from each section, 2 marks total. Do not allow "while names are in the wrong order". BP4 must have reference to checking a condition / condition being met, not just having a condition.

Question			Answer	Mark	Guidance
			• More efficient than / does not need to iterate as many times as **count controlled / for loop**		
3	(c)	(i)	1 mark each for insertion and bubble sort, max 2 Insertion sort: • inserts/moves values into **correct position** • inserts value **once** (then in correct position) • stops when end of array reached // completes in one **pass** through the array • moves items down the array / left • start of array becomes sorted first • creates a sorted array **within** an array // has a sorted/unsorted **partition / section / list** • starts on 2nd value • more efficient/faster than bubble sort • … because fewer iterations / comparisons (on average) • … when data more scrambled Bubble sort : • compares/swaps **pairs** of values • value is **repeatedly** moved/swapped (until in correct position) • repeats if a swap has been made // needs multiple passes	**2** (AO1 1b)	Answer must reference both bubble sort and insertion sort for 2 marks except if efficiency mark plus expansion given. Allow reference to big O for efficiency discussion. Only award efficiency once. Only award fewer iterations once Do not accept “completes in one iteration” for insertion sort. Accept list / data / values / etc for array. “when data more scrambled” only makes sense when discussing efficiency/speed, do not give marks for saying that either can handle data that is more scrambled (they both can sort data however it is arranged). Do not accept “bubble/insertion sort does not” for 2nd mark.

Question			Answer	Mark	Guidance
			• will complete a final iteration once sorted (to check for no swaps needed) • moves items up the array • end of array becomes sorted first • moves/bubbles the highest value to the top • less efficient/slower than insertion sort (on large sets of values) • … more iterations / comparisons (on average) • … when data more scrambled		
3	**(c)**	**(ii)**	1 mark each to max 2 e.g. • Both produce a sorted list / array • Both work in place / without duplicating data / without using divide and conquer • Both need a temporary variable • Both swap values • Both use loops / iteration / repeats • Both loops are nested / inside each other • Both (may) need multiple passes • Both use selection • Both work with an array / list data structure • Both work from left to right • Both build up **sorted list** one item at a time (after every pass) • Both compare (pairs of) **values** • Both (typically) **less efficient / slower** than merge sort (or other sorting algorithms)	**2** (AO1 1b)	Allow reference to both sorting / putting items into order for BP1. “Allows sorting of numbers and strings” meets BP1 Allow answers relating to not needing additional memory as BP2. Allow “breaking into smaller lists” as divide and conquer for BP2. If answer is a statement (e.g. “uses loops”), assume candidate is talking about both algorithms doing this.

Question			Answer	Mark	Guidance
			• Both inefficient / slow for **larger / unsorted** lists **//** efficient for **small** / (nearly) **sorted** lists • Both start by comparing first two values		

Original question 3

The full source mark scheme begins on page 45. Use the original paper code, tier and question number shown above.

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.

<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>

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.

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)

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.

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.

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.

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			Answer	Mark	Guidance
3	(a)	(i)	• Split up into individual lists each of **one item** (divide stage) • **[2,7] [1,3] [6,9] [4,5]** (sorted lists of 2) • **[1, 2, 3, 7] [4, 5, 6, 9]** (sorted lists of 4) • **[1, 2, 3, 4, 5, 6, 7, 9]** after reasonable attempt at merging from **4 to 8**	4	**Do not allow merging out of order and <u>then</u> sorting for BP 3 or 4.** Splitting can be done in multiple stages or all at once. If no attempt at splitting up, assume initial list is already split up and mark BP2, 3 and 4. If splitting done incorrectly, do not give BP1 but allow access to further points. Do not accept sorted list for BP4 if no steps shown or wrong algorithm used. Brackets not needed around numbers – must be reasonably clear numbers are in 2s, 4s, etc. If candidate uses numbers other than the ones given in the question or sorts into descending order, penalise once then FT.
3	(b)		• Uses an array of numbers with a sorted part and unsorted part **(top box)** • Swaps them if they are in the incorrect order **(bottom box)** • Is declared with a fixed length **(top box)**	3	No mark if more than one box ticked per section.
3	(c)	(i)	• Linear search	1	Accept ”linear” by itself Do not allow “linear **sort**”

Question			Answer	Mark	Guidance
3	(c)	(ii)	• End of list reached // gets to last value (and not found) // checked every value (and value not found)	1	Do not accept "value not found / value not in list" by itself. Must have idea of getting to the end or having checked the entire list.

PMT

GCSE (9–1)

Computer Science

J276/02: Computational thinking, algorithms and programming

General Certificate of Secondary Education

Mark Scheme for June 2019

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 2019

PMT

J276/02 **Mark Scheme** **June 2019**

Annotations

Annotation	Meaning
BP	Blank page
^	Omission mark
BOD	Benefit of doubt
	Cross
FT	Follow through
NAQ	Not answered question
NBOD	Benefit of doubt not given
REP	Repeat
	Slash
	Tick
TV	Too vague
0	Zero (big)
SEEN	Noted but no credit given

PMT

J276/02 **Mark Scheme** **June 2019**

Question			Answer	Mark	Guidance
1	(a)	(i)	1 mark per bullet to max 2. • **Height/amplitude** of waveform is sampled/measured • Converted to / stored as binary/digital • Sample / measurements taken at a **regular interval** / **set interval** / by sensible example (eg 44,000 times per second)	2 AO1 1b (2)	Do not accept frequency Do not accept unrealistic sample rates (e.g. once per second).
1	(a)	(ii)	1 mark per bullet to max 1. • number of samples taken **per second / per time period** • How <u>often/regularly</u> a sample is taken	1 AO1 1a (1)	Accept reference to Hertz (Hz) as time period.
1	(a)	(iii)	1 mark per tick to max 2. Tick (✓) **two** boxes The file size of the digital recording will be smaller The file size of the digital recording will be larger ✓ The quality of playback of the digital recording will be better. ✓ The quality of playback of the digital recording will be worse.	2 AO1 1b (2)	If 3 ticks given, max 1 mark If 4 ticks given, 0 marks.
1	(b)	(i)	1 mark per bullet to max 3. • Image **made of / split up** into **pixels** • Each pixel given a binary code… • …which represents the colour of that pixel • Each colour is given a **different/unique** binary code. • Metadata stored alongside the image	3 AO1 1b (3)	BP1 needs idea of picture made up of pixels, not just mention of the word “pixel” Not enough to say “each colour is given a binary code”, must have the idea of this being unique or different for each different colour. Accept examples of metadata such as height/width, geolocation, etc. Do not accept file size/file name.

PMT

J276/02 **Mark Scheme** **June 2019**

1	**(b)**	**(ii)**	1 mark per bullet to max 2. • Computers consist of transistors / switches / logic circuits / gates… • …which only have two values / on or off / 1 or 0 / open or closed	2 AO1 1b (2)	Only give BP2 if BP1 given. BP1 must refer to **hardware** that switches between two states. Do not accept processor for this.
1	**(b)**	**(iii)**	1 mark per bullet 2 marks max for advantages 2 marks max for disadvantages Advantages • **File** size is reduced / gets smaller // image contains less data • ..so quicker to **upload / download / load / transfer** • ...takes up less storage space // space on the web server • …less (mobile) data usage • …permanent reduction Disadvantages • Quality of image is reduced (compared to original) • …because data is lost / removed // by example (eg fewer colours / lower resolution) • …reduction is permanent / not reversible	4 AO2 1b (4)	Do not accept “size” to mean “file size” for advantage. “Data is permanently removed” gets 2 marks for a disadvantage

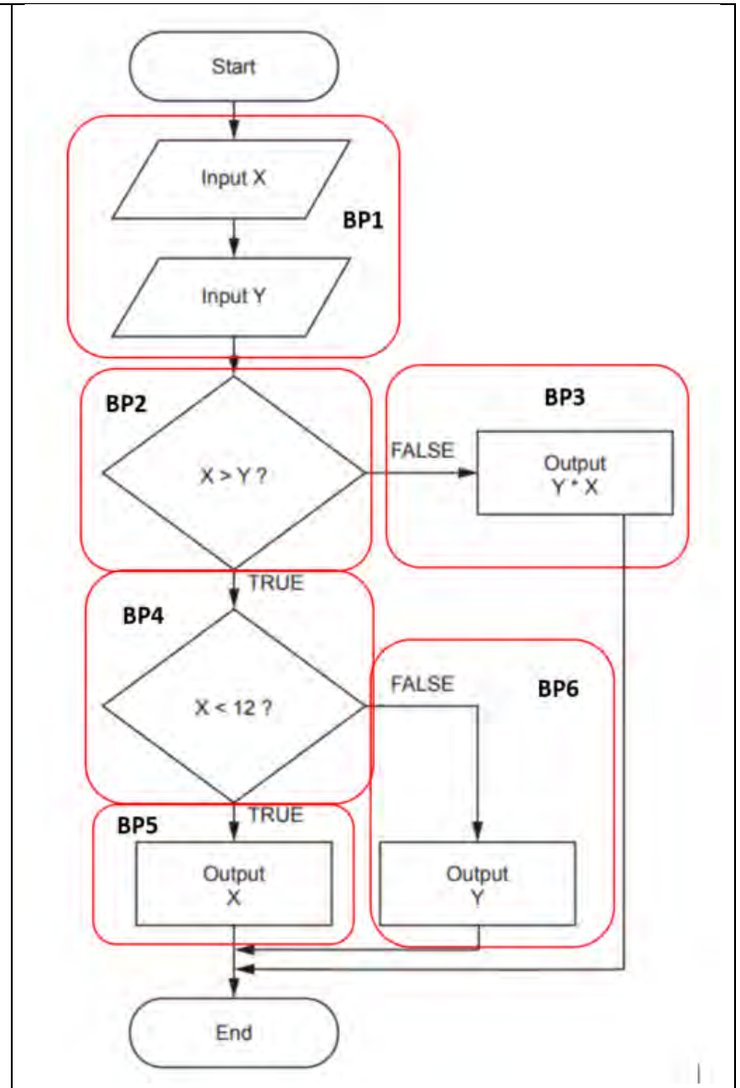
PMT

J276/02 **Mark Scheme** **June 2019**

2	**(a)**		1 mark per bullet to max 4, 1 mark per row • 10 • 6 • 6 • 2	4 AO2 1b (4)	Correct Answer Only Do not accept “X”, “Y”, etc.

2	**(b)**		1 mark per bullet to max 6.
			
			• Inputs two value (as X and Y)
			• Compares if X is larger than Y…
			• …Outputs Y*X only when <u>False</u>
			• Compares if X is less than 12…
			• …Outputs X only when True <u>and X > Y</u>
			• …Outputs Y only when False <u>and X > Y</u>
				6
				AO3 2b (6)
					Question specifically asks for pseudocode.
					
					Outputs should only be given if they occur with the right condition(s).
					
					<u>Example algorithm</u>
					
					`input x`
					`input y`
					`if x > y then`
					`  if  x < 12 then`
					`  print x`
					`  else`
					`  print y`
					`  end if`
					`else`
					`  print y*x`
					`end if`
					
					Variables do not have to be called x and y.
					
					Accept equivalent comparisons (e.g. `if X <= Y`)
					
					Allow FT for outputs from incorrect comparisons where a sensible attempt has been made.

J276/02 **Mark Scheme** **June 2019**

			Please note how mark bullet points match up to the flowchart given in the question. BP1 is for taking both inputs BP2 and 4 are for correct comparisons of variables. This may be done in alternative ways (e.g. X<=Y, X>=12, etc) BP3, 5 and 6 are for the correct outputs in the right place. If the answer logically works to produce the correct output, it should be marked as correct.		

5

J276/02 **Mark Scheme** **June 2019**

2	**(c)**		1 mark per bullet to max 4, 2 mark max per method • Compiler • …translates code **in one go** / all at once • …produces an executable file // does not need to be compiled again • Interpreter • …translates code line by line. • …will be interpreted / translated every time it is run.	4 AO1 1b (4)	Mark first method only in each section
3	**(a)**	**(i)**	1 mark per bullet to max 1 • An error that results in **incorrect output / unexpected result** • Contains an error but **still runs / doesn't crash**	1 AO1 1b (1)	Do not accept examples of logic errors.

3	**(a)**	**(ii)**	`if num MOD 2 == 0 then`
			
			`if num MOD 2 = 0 then`
				1
				AO3 2b (1)
					Important point is that >= is changed to == or =.
					Accept alternatives that produce the same result (e.g. <=0, <1, !=1, etc.)
					
					Ignore any casting (e.g. using int() to convert to a number)
					
					Accept other minor changes to the line as long it logically works.
					
					Accept versions of MOD from high level languages (e.g. Python : `if num % 2 == 0`)

J276/02 **Mark Scheme** **June 2019**

3	**(b)**	**(i)**	1 mark per bullet to max 1 • An error in the grammar of the program // error that breaks the rules of the programming language • Contains an error but will not run / translate / execute	1 AO1 1b (2)	**Do not** accept examples of syntax error (e.g. misspelling).
3	**(b)**	**(ii)**	`print("odd")`	1 AO2 1b (1)	Must include quotes (single or double). Do not penalise spelling mistakes in message. Accept sensible alternatives to "odd" Accept alternatives for print / output as long as spelling is accurate
4	**(a)**	**(i)**	1 mark per bullet to max 2 • Removing / hiding / obscuring **unnecessary** detail • Focusing on the important detail • Simplifies **the problem** // reduces complexity // Easy to solve / understand	2 AO1 1a (1) AO1 1b (1)	Accept answers relating to using fewer computational resources Must be the programmer making the decision.
4	**(a)**	**(ii)**	1 mark per bullet to max 1 • Suitable example of what can be focused on (e.g. player name, match results, goals scored) • Suitable example of what to remove/hide (anything relevant that is not results/goals scored) • Suitable example of a simplification made	1 AO2 1a (1)	Mark first answer only Allow any suitable example of abstraction as long it is relevant to the system. Allow either first name or surname to be removed as an example, but do not allow both to be removed.

J276/02 **Mark Scheme** **June 2019**

4	**(b)**		1 mark per bullet, mark in pairs. Max 2 per point. e.g. • Input sanitisation • …cleaning up input data / removing unwanted data • …by example (e.g. removing special characters / preventing SQL injection) • Validation • …checking whether input data should be allowed / is sensible / follows criteria • …by example (e.g. goals cannot be less than 0) • Verification • … checking whether data has been entered correctly • …by example (e.g. double entry / visual check) • Authentication • …ensuring only allowed / authorised users can gain access • …by example (e.g. usernames /passwords) • Maintainable code • …to allow other programmers to understand the code • …by example(e.g. comments, indentation, meaningful variable names)	4 AO2 1a (2) AO2 1b (2)	Mark first answer only in each section For validation, allow one example of a type of validation (e.g. type check, range check) e.g. question so allow other sensible examples such as audit logging, encryption of data Do not allow “data is correct” as expansion for validation – validation checks that data is sensible or follows rules, NOT that it is correct. Planning for contingencies and anticipating misuse are not examples by themselves, but discussion of these may fit under other points – e.g. input sanitisation, validation.

PMT

J276/02 **Mark Scheme** **June 2019**

4	**(c)**		1 mark per bullet to max 3 • `count` • `= nogoalscount + 1` • `nogoalscount`	3 AO3 2b (3)	Correct answer only. Accept alternatives to adding 1 to variable (e.g. `+= 1` / `++`) Penalise spelling once only, FT for further mistakes. Do not penalise case. Accept sensible messages printed out alongside `nogoalscount`
5	**(a)**		1 mark per bullet to max 2 • 163 • **Correct** working shown.	2 AO1 1b (2)	Award working mark independently of final answer but working must be correct (e.g. (16 x 10) + 3)
5	**(b)**		1 mark per bullet to max 2 • 91 • **Correct** working shown.	2 AO1 1b (2)	Award working mark independently of final answer but must be correct (e.g. 1+2+8+16+64 // correct binary headings with correct binary underneath)
5	**(c)**		• 9	1 AO1 1b (1)	Correct answer only Do not accept 3^2 or 3 x 3
5	**(d)**		1 mark per nibble to max 2 • 1101 1101	2 AO1 1b (2)	Mark from right to left.

PMT

J276/02 **Mark Scheme** **June 2019**

5	**(e)**		1 mark per missing bit **A** | **B** | **Q** **0** | **0** | **0** **0** | **1** | **1** **1** | **0** | **1** **1** | **1** | **1**	4 AO2 1b (4)	Accept T / True

10

PMT

J276/02 **Mark Scheme** **June 2019**

6	**(a)**	**(i)**	1 mark per bullet to max 6
			
			• Function ticketprice() **defined**
			• … that accepts two parameters and has no other inputs
			• Works out total ticket price for adult (eg adult * 19.99)
			• Works out total ticket price for children (eg child * 8.99)
			• Adds on **correct** booking fee
			• Returns the calculated value.
				6
				AO3 2b (6)
					Bullet points 3, 4, 5 can be awarded even if no mention of a function / parameters (for example, if candidate has inputted the number of tickets needed.
					
					Do not award return value if no attempt at a function. Return mark can be given if a good attempt made at calculating the total, even if this is incorrect.
					
					Allow 2.50 booking fee to be per order or per ticket
					
					Ticket prices must be stored appropriately if needed.
					
					example algorithm
					
					`function ticketprice(numadult, numchild)`
					`  price = (numadult * 19.99) + (numchild * 8.99) + 2.50`
					`  return price`
					`end function`
					
					Allow alternatives in high level languages (e.g. def in Python).
					
					Allow return as assigning the value to the name of the function (VB syntax)

6	**(a)**	**(ii)**	1 mark per bullet to max 2 • Real… • …Returned value may not be a whole number / may have a decimal point in	2 AO2 1a (1) AO2 1b (1)	Allow String only if matching justification shows understanding (e.g. £ sign attached, message returned alongside value).

J276/02 **Mark Scheme** **June 2019**

6	**(b)**	**(i)**	1 mark per bullet to max 1 • Check that the code is valid / real • Check it has been entered / sent / received correctly. • Makes it harder for people to make up discount codes	1 AO2 1b (1)	Mark first answer only
6	**(c)**	**(i)**	• Not in order / sorted	1 AO2 1b (1)	Mark first answer only
6	**(c)**	**(ii)**	• Linear (search)	1 AO1 1b (1)	Mark first answer only Allow other valid searching algorithms as long as they work on an unsorted list (e.g front and back search)
6	**(d)**	**(i)**	1 mark per bullet to max 2 • Flag / record whether a swap has taken place or not • checked as condition to decide **whether to repeat**	2 AO2 1b (2)	The variable records whether a swap has taken place; it does **no**t perform the swap.
6	**(d)**	**(ii)**	1 mark per bullet to max 2 • Swaps.. • …values of `queuesize[p]` and `queuesize[p+1]` • …when `queuesize[p]` is larger than `queuesize[p+1]` • using a temporary variable //doesn't overwrite numbers //explanation of process	2 AO2 1b (2)	Do not accept "sorts numbers" "swaps numbers" meets BP1. Explanation of which values in the array are swapped meets BP1 and BP2. Do not accept direct word for word repetition from the program (e.g. `temp = queuesize[p]`), question asks for an explanation. Explanation of temporary variable must be logically correct.

J276/02 **Mark Scheme** **June 2019**

6	**(d)**	**(iii)**	1 mark per bullet to max 2. • Comments • … to enable programmers to understand the purpose of each line / section • …by example (e.g. on line 4 add the comment…) • Naming variables sensibly • … to enable programmers to understand the purpose of each variable • …by example (e.g. change identifier p to …) • Modularise • …to allow reuse / makes easier to test / reduces errors • …by example (e.g. create as a function)	2 AO2 1a (1) AO2 1b (1)	Mark first answer only Do not accept indentation (already done) Accept “show what each line does” for comments.
6	**(d)**	**(iv)**	1 mark per bullet to max 2. • Insertion (sort) • Merge (sort)	2 AO1 1a (2)	Accept “insert”. Do not penalise spelling. Do not accept bubble sort (given in previous questions) Do not award searching algorithms Allow other valid sorting algorithms. (e.g. quick sort, heap sort, shell sort, selection sort, radix sort, bucket sort, tim sort, comb sort, pigeonhole sort, etc.)

J276/02 **Mark Scheme** **June 2019**

6	**(e)**		1 mark per bullet to max 8.
			
			• Input height
			• Accepts riders > / >= 140 with suitable message
			• Rejects riders < / <= 120 with suitable message
			• Checks if height between 120 and 140…
			• … If True, input whether accompanied
			• … Suitable output message for True AND False
			• Correctly counts number of riders <u>in all cases of being allowed to ride</u> *(do not penalise candidates for counting or not counting accompanying adults)*
			• Attempt to loop based on 8 riders allowed
			
			Some checks for rider height may be implicit (e.g. using ELSE after checking other heights). If the answer logically works to produce the correct output, it should be marked as correct.
			
			Loop will almost certainly be condition controlled (WHILE/DO UNTIL) to gain BP8; count controlled (FOR) loop requires significant manipulation to work successfully.
				8
				AO3 2b (8)
					Answers can be in any suitable format (including pseudocode, flowchart, etc). If flowchart used, accept any sensible shapes.
					
					Do not penalise for lack of initialisation of variables.
					
					Loop must repeat until 8 riders allowed, not just loop 8 times.
					
					Do not credit asking whether accompanied if in the wrong place.
					
					Condition for BP4 may be `120 < h < 140`
					
					<u>Example algorithm</u>
					`riders=0`
					`while riders <8`
					`  input height`
					`  if height >= 140 then`
					`  output "allowed"`
					`  riders = riders + 1`
					`  elif height >=120 then`
					`  input withadult`
					`  if withadult == "yes"`
					`  output "allowed"`
					`  riders = riders + 1`
					`  else`
					`  output "not allowed"`
					`  end if`
					`  else`
					`  output "not allowed"`
					`  end if`
					`endwhile`

PMT

OCR (Oxford Cambridge and RSA Examinations)
The Triangle Building
Shaftesbury Road
Cambridge
CB2 8EA

OCR Customer Contact Centre

Education and Learning
Telephone: 01223 553998
Facsimile: 01223 552627
Email: general.qualifications@ocr.org.uk

www.ocr.org.uk

For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored

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 (Oxford Cambridge and RSA Examinations)
Head office
Telephone: 01223 552552
Facsimile: 01223 552553

© OCR 2019

PMT

OCR
Oxford Cambridge and RSA

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

Oxford Cambridge and RSA Examinations

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.)*

J277/02 Mark Scheme June 2024

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.

J277/02 Mark Scheme June 2024

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

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:

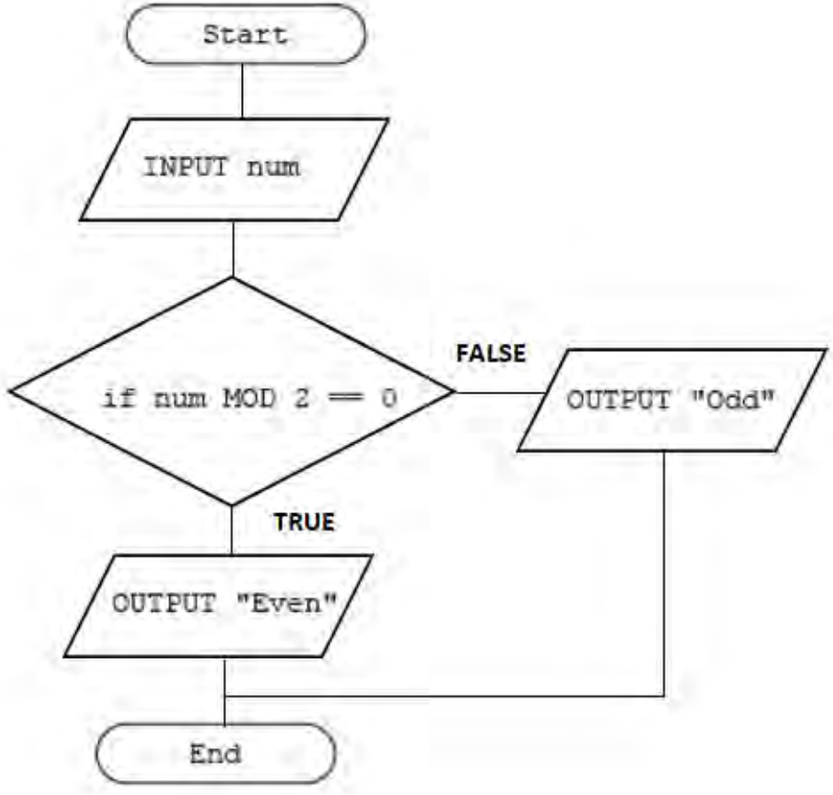
- Blank pages at the start of the script need SEEN annotation
- Any questions answered elsewhere (e.g. on the first blank pages, separately on the page) need to be linked within RM Assessor and annotated with ticks/crosses/SEEN as appropriate
- 1 tick for every mark awarded, if a question is given 3 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				3 (AO1)	

Keyword	Programming construct	
	selection	iteration
`if`	✓	
`for`		✓
`while`		✓

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")

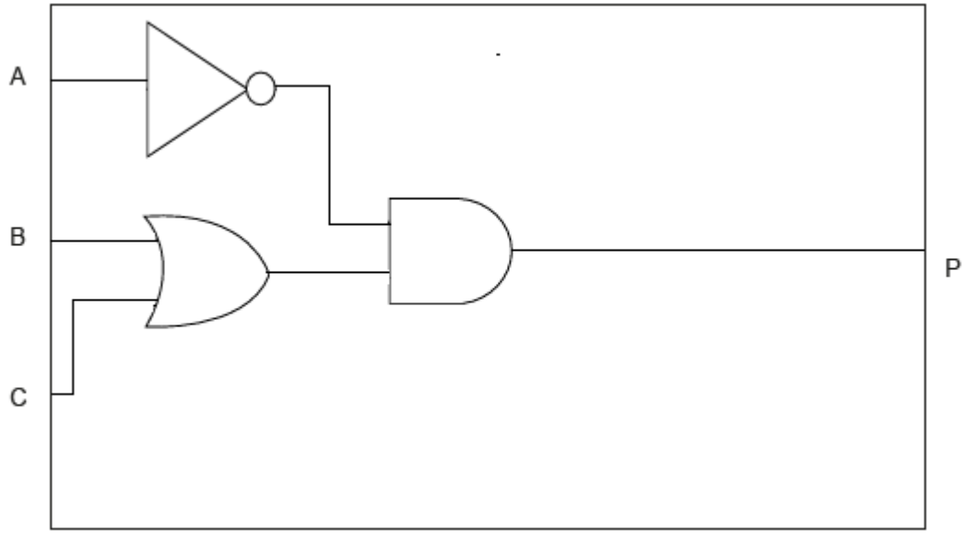
4	**(b)**		Only 1 method asked for. Could be name and description/example or description and example • Authentication • …checking users allowed to access the site / know identity of users • … by example (e.g. username and password) • Anticipating misuse // preventing misuse • ….stopping the user breaking / 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.

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 <code>word1</code> and <code>word2</code>
• correct concatenation (<code>word1</code> then <code>word2</code>)...
• …storing in variable <code>message</code>

Example
<pre>word1 = "Hello"
word2 = "Everyone"
message = word1 + word2</pre></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>

J277/02 Mark Scheme June 2024

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.

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
			
			Example 1
			`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)`
			
			Example 2 (alternative)
			`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.

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) Tick (✓) **one** box in each row to identify whether the OCR Reference Language code given is an example of selection or iteration.

OCR Reference Language code	Selection	Iteration

`for i = 1 to 10`
`  print(i)`
`next i`		

`while score != 0`
`  playgame()`
`endwhile`		

`if playerHit() then`
`  score = 0`
`endif`		

`switch bonus:`
`  case 0:`
`  score = 9`
`  case 1:`
`  score = 7`
`  case 2:`
`  score = 5`
`endswitch`		

[4]

Candidates appeared to struggle with this question. In particular the use of switch/case was not well understood. This may be because some high-level languages such as Python have not traditionally supported this. The recent update to Python 3.10 introduces this construct. Centres who use Python may want to consider upgrading their installation to allow practical implementation of the switch/case construct.

Question 1 (b)

(b) Write pseudocode to increment the value held in the variable `score` by one.

.....

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

Most candidates were able to temporarily add one to the value of score (by using code such as `score + 1`). Fewer were able to assign this new value to the variable of score and therefore fully answer the question. Answers such as `score = score + 1`, `score++` or `score += 1` were all accepted, as were any longer responses that used intermediate variables. As long as the value of `score` ended up being incremented by one, examiners were instructed to give credit.

Assessment for learning

One notable response that was **not** allowed was `score =+1`. This statement assigns the value of 'positive 1' to score, overwriting the previous value held.

This is a good example of the precision required to gain marks. Candidates with practical programming experience are more likely to recognise this.

Question 1 (c)

(c) State the name of each of the following computational thinking techniques.

Breaking a complex problem down into smaller problems.

.....

.....

Hiding or removing irrelevant details from a problem to reduce the complexity.

.....

.....

[2]

This question was answered well by the majority of candidates.

Question 3 (a)

3 A program stores the following list of positive and negative numbers. The numbers need sorting into ascending order using a merge sort.

45	12	–99	100	–13	0	17	–27

(a) The first step is to divide the list into individual lists of one number each. This has been done for you.

Complete the merge sort of the data by showing each step of the process.

45	12	–99	100	–13	0	17	–27

[3]

Most candidates were able to correctly demonstrate the required steps of the merge sort algorithm. This involves merging the set of 8 individual lists into 4 lists of size 2, then 2 lists of size 4, then 1 list of size 8. At each point, each list produced must be in order. It is this process of merging lists together that sorts the values.

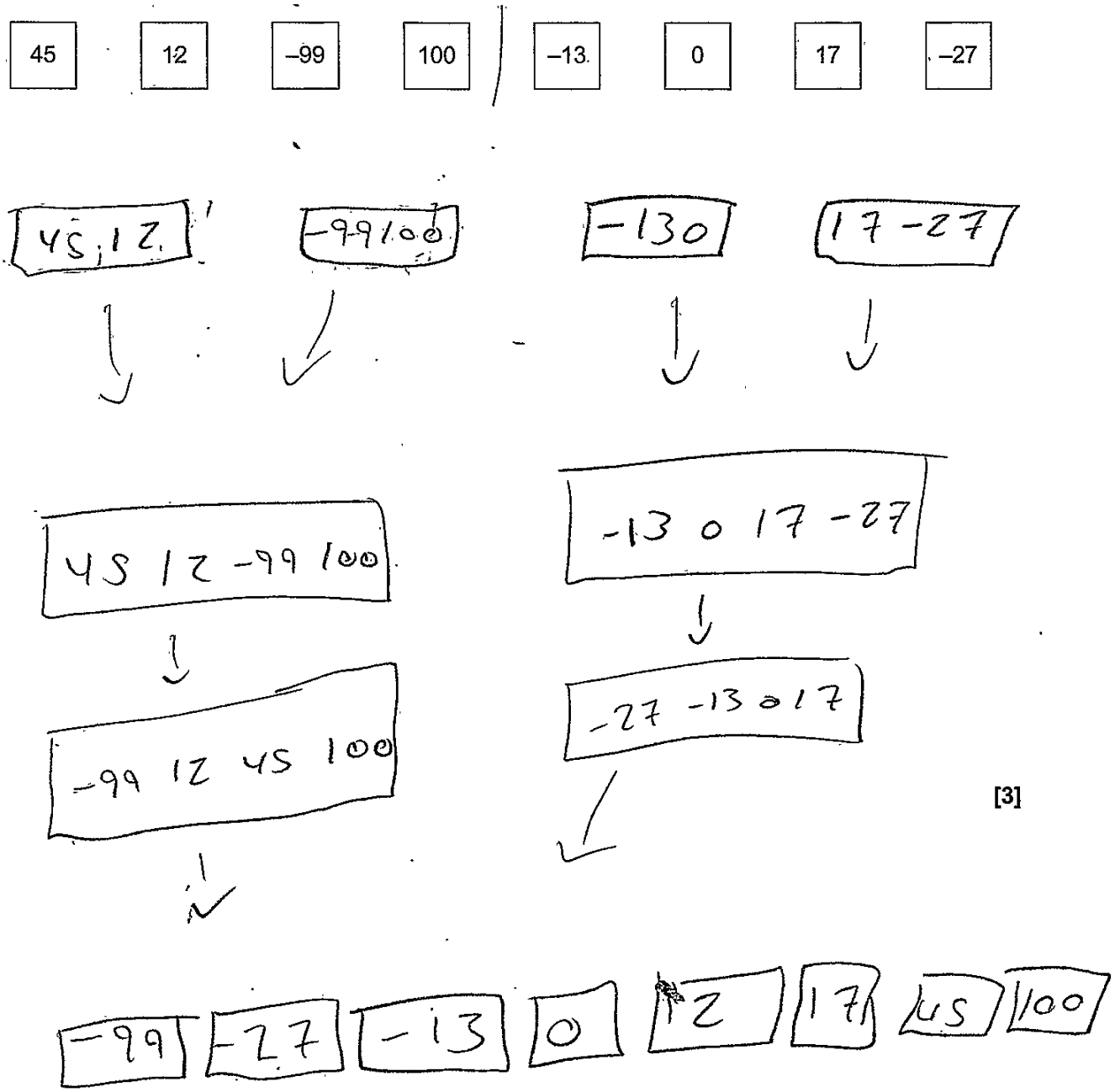
No mark was given simply for showing the final list sorted unless the previous step(s) had been followed; any candidates an alternative sorting algorithm (for example a bubble sort) would achieve 0 marks even though the final outcome is the same.

Misconception

Many candidates showed merged lists which were out of order to start with, and then sorted these merged lists. This is highly inefficient and not what the standard merge sort algorithm describes.

In merge sort, the process of merging lists together produces the sorted list. There does not need be any second step to sort values as they should already be in order.

Exemplar 2



[3]

This exemplar shows the misconception detailed previously. Here, the candidate firstly merges together the individual lists into 4 lists of size 2, but these lists are not all in order (e.g. 17, -27 is incorrect).

The second step then merges these lists together into 2 lists of size 4, but again, these are not in order. An unnecessary next step is then shown where the lists are then sorted in place.

Question 3 (b)

(b) Once the numbers are in order, a binary search can be run on the data.

Describe the steps a binary search will follow to look for a number in a sorted list.

.....

.....

.....

.....

.....

..... **[4]**

This question asks candidates to describe how a binary search would look for a number in a sorted list. Most candidates answered this well. The most common mark for this question was 3 out of 4.

The majority of candidates clearly understood that the middle value is chosen and compared against the value being searched for. However, almost all candidates then went on to discuss which half of the list should be discarded or focused on based on this comparison.

One possibility is that the middle value is the number we are searching for. Few candidates discussed this possibility. As the process repeats, this check is important if the number is ever to be found!

Assessment for learning

When discussing a binary search there are **three** possibilities when comparing the middle value to the number being searched for.:

1. the middle value is higher than needed (in which case the left half of the list should be focused on)
2. the middle value is lower than needed (in which case the right half of the list should be focused on)
3. the middle value is equal to the value needed (in which case the value has been found and the process can stop.

Most responses only considered the first two of these possibilities. This is worth discussing when teaching this topic.

Question 3 (c)

(c) A linear search could be used instead of a binary search.

Describe the steps a linear search would follow when searching for a number that is **not** in the given list.

.....

.....

.....

..... **[2]**

Responses to this question generally lacked precision.

The question specifically asks about searching for a number that is **not** in the list. However, many candidates discussed stopping when the value had been found. This is clearly not possible given the scenario.

Many responses discussed checking each number but did not state whether this would be in a particular order. Some candidates discussed checking values randomly, or even from the right down to the left.

For this questions, candidates may have known that a linear search starts at the left most number and then checks each value in the list in order until reaching the end of the list. However, few responses stated this or anything close to this.

This level of precision (and conversely, not assuming that steps are obvious) should be encouraged within GCSE Computer Science.

Question 3 (a)

3 An insertion sort is one type of sorting algorithm.

A student has written a pseudocode algorithm to perform an insertion sort on a 1D array `names`.

```
names = ["Kareem", "Sarah", "Zac", "Sundip", "Anika"]
for count = 1 to names.length - 1
    pos = count
    while (pos > 0 and names[pos] < names[pos - 1])
        temp = names[pos]
        names[pos] = names[pos - 1]
        names[pos - 1] = temp
        pos = pos - 1
    endwhile
next count
```

(a) Describe the purpose of the variable `temp` in the insertion sort pseudocode algorithm.

.....

.....

.....

..... **[2]**

This question was an excellent discriminator in terms of candidate achievement. Many responses correctly described the `temp` variable as storing the name so that it could be overwritten during the process of swapping values. Partial credit was given for responses simply commenting on it storing data, as this is essentially the purpose of any variable.

However, credit was not given where candidates discussed storing a position/index, as this was not the case; the `pos` variable is used as the index of the array. The content of this array index (a name) is stored in the `temp` variable.

Where responses are open to interpretation, the mark scheme attempts to credit as many of these as possible. However, incorrect responses (such as reference to the position here instead of the name) are not credited.

Question 3 (b)

(b) An insertion sort contains a nested loop; a loop within a loop. In this pseudocode algorithm the outer loop is a count-controlled loop and the inner loop is a condition-controlled loop.

Explain why the inner loop needs to be a condition-controlled loop.

.....

.....

.....

..... **[2]**

Previous questions such as Question 2(a) assessed knowledge (AO1). This question focused on the application (AO2) of a technique (condition-controlled loops) to the algorithm given (an insertion sort). This assessed candidates' understanding of the process an insertion sort follows to sort values.

The J277 specification is clear that candidates do not have to remember the code for this algorithm. But the specification does state that candidates must be able to understand the main steps of the algorithm and identify the algorithm if given the code for it. In this case, candidates were given all of the code for the algorithm.

Candidates generally found this question challenging. Many responses simply repeated the question and discussed sorting values. More successful responses understood that the inner loop is the part of the algorithm that moves the `name` repeatedly in the list and that we do not know how many times this move needs to be made. Further to this, it is the condition that the `name` is in the correct position (or even better, an explanation of how this is decided on) which ends the loop.

Assessment for learning



When asked to explain why a particular technique is used, it is often useful for candidates to think about why alternative options have **not** been used. In this question, thinking about what would happen if a count-controlled loop had been used as the inner loop may have given candidates the insight to discuss why a condition-controlled loop has been used.

Question 3 (c) (i) and (ii)

(c) A bubble sort is another type of sorting algorithm.

(i) Describe **one** difference between an insertion sort and a bubble sort.

.....

.....

.....

..... **[2]**

(ii) Describe **two** similarities between an insertion sort and a bubble sort.

1 .....

.....

2 .....

.....

[2]

Question 3(c)(i) and Question 3(c)(ii) asked candidates to describe one difference and two similarities between an insertion sort and a bubble sort.

Examiners were instructed to be generous in their interpretation of the requirements of both algorithms. This included considering both their algorithmic implementation and the understanding of how these are typically described on a classroom whiteboard.

As such, many combinations of answers could gain the marks available.

Question 3 (a)

3

(a) State what is meant by the term syntax error. Give **one** example of a syntax error in a program.

Definition .....

.....

Example .....

.....

[2]

This question firstly asked for a definition of the term 'syntax error'. Although many candidates were correctly able to do this in terms of rules of the programming language being broken, many instead gave examples of issues that would cause syntax errors, such as 'where a bracket is missing'. This would indeed cause a syntax error in many high-level languages but is not a definition in general and so was not credited by examiners for this part of the question.

The second part did ask for an example and generous interpretation was asked of examiners so that any issue that could feasibly cause a syntax error was awarded, such as missing brackets or misspelling of key words. One common misconception here involved missing quotation marks/string delimiters around a string, which could instead be a reference to a variable if this was a single word.

Misconception

A syntax error is a mistake with the rules/grammar of the programming language that means the program will not run/execute or compile.

Code such as `print(temp)` would not necessarily be a syntax error because `temp` could plausibly be a variable.

Question 3 (b)

(b) A student writes an algorithm to input two numbers and add them together to create a total.

If the total is between 10 and 20 inclusive, "success" is output.

If the total is not between 10 and 20 inclusive, "warning" is output.

```
01 num1 = input("Enter a number")
02 num2 = input("Enter a number")
03 total = num1 + num1
04 if total >= 10 then
05     print("success")
06 else
07     print("warning")
08 endif
```

The algorithm does not work correctly.

Identify the line number of the **two** logic errors in the algorithm and refine the code to correct each logic error.

Line number .....

Correction .....

.....

Line number .....

Correction .....

.....

[4]

Line 03 was commonly identified as containing a logic error and many candidates were able to correct this. Where a candidate attempted to explain what changes should be made, this was only credited where the explanation was unambiguous and precise.

The correction to line 04 was commonly done incorrectly due to the need to check multiple values.

Misconception

Where multiple values are required to be checked in a selection statement, a line such as :

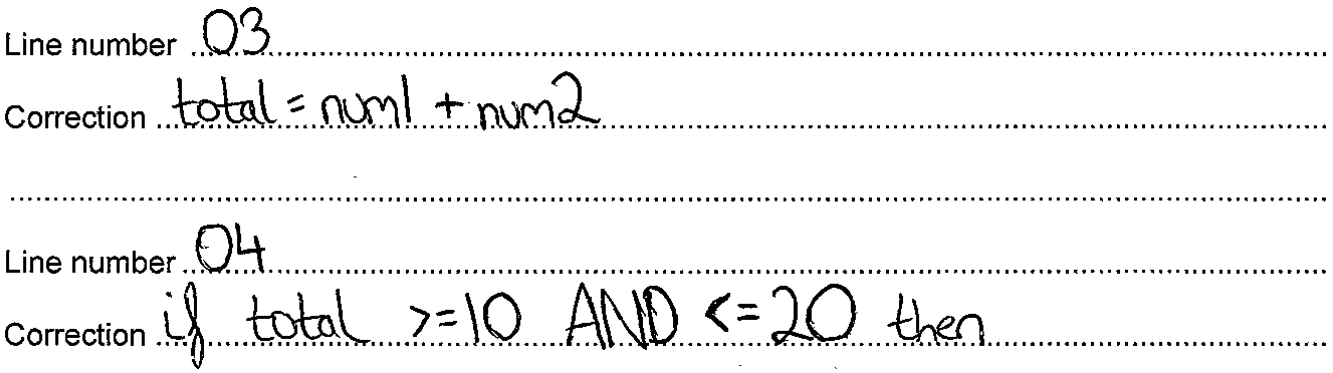
```
if total >= 10 and <= 20 then
```

is incorrect as the second part of the statement has nothing to compare 20 against. The first part will clearly evaluate to `true` or `false`, but the second part is ambiguous. Instead, candidates should be encouraged to use :

```
if total >= 10 and total <= 20 then
```

Alternatives that work in high-level languages would also obviously be accepted.

Exemplar 1



Although this candidate has correctly identified lines 03 and 04, the correction for line 04 is incorrect due to the missing reference to total when comparing the upper boundary. This achieves three marks out of four.

Question 9 (a) (i)

9 Students take part in a sports day. The students are put into teams.

Students gain points depending on their result and the year group they are in. The points are added to the team score.

The team with the most points at the end of the sports day wins.

(a) Data about the teams and students is stored in a sports day program.

(i) Identify the most appropriate data type for each variable used by the program.

Each data type must be different.

Variable	Example	Data type
`teamName`	`"Super-Team"`	
`studentYearGroup`	`11`	
`javelinThrow`	`18.2`	

[3]

This question was completed very well by the vast majority of candidates, showing that the use of data types are now well understood by centres.

Question 9 (a) (ii)

(ii) The student names for a team are stored in an array with the identifier `theTeam`

An example of the data in this array is shown:

Index	0	1	2	3	4	5
Data	Ali	Eve	Ling	Nina	Sarah	Tom

theTeam

A linear search function is used to find whether a student is in the team. The function:

- takes a student name as a parameter
- returns `True` if the student name is in the array
- returns `False` if the student name is **not** in the array.

Complete the design of an algorithm for the linear search function.

```
function linearSearch(studentName)

    for count = 0 to ..........

        if theTeam[..........] == .......... then

            return ..........

        endif

    next count

    return False

endfunction
```

[4]

This was a challenging question revolving around the use of an array that is iterated through to implement a linear search. Many candidates achieved two marks for the first and last points, understanding that the loop would repeat from indexes 0 to 5 and that the value `True` would be returned if the item was found. As usual, allowance was given for off-by-one errors with this loop because of the prevalence of Python in centres and how loops are count controlled loops are handled in this language.

It was far less common for candidates to achieve the middle two marks, perhaps because the level of technical knowledge needed was greater. A number of candidates attempted here to fashion a 2D array and refer to multiple indexes, but this was not appropriate given the data structure given. The most challenging mark was certainly the use of `count` as the index of the `theTeam` array, with very few candidates correctly identifying this as the missing element.

Assessment for learning



Centres are encouraged to link the topics of arrays (both single and two-dimensional) to count controlled loops to be confident in answering questions like this one.

A very common programming exercise is to iterate through every item in an array, either to search for an item, count or add items or as the pre-cursor for a search. Candidates are expected to have significant practical programming experience over the duration of their studies.

Question 9 (b)

(b) This algorithm calculates the number of points a student gets for the distance they throw in the javelin:

```
01   javelinThrow = input("Enter distance")
02   yearGroup = input("Enter year group")
03   if javelinThrow >= 20.0 then
04       score = 3
05   elseif javelinThrow >= 10.0 then
06       score = 2
07   else
08       score = 1
09   endif
10   if yearGroup != 11 then
11       score = score * 2
12   endif
13   print("The score is", score)
```

Complete the trace table for the algorithm when a student in year 10 throws a distance of 14.3

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

Line number	javelinThrow	yearGroup	score	Output

[4]

Trace tables have appeared multiple times in J277 examination papers now and candidates are hopefully familiar with the expectations, which are consistent across series.

Most candidates correctly traced through the given algorithm, which was more accessible than usual due to not containing any loops. Where mistakes were made, this was typically to do with either incorrect line numbers being given for each change (this was penalised once only and then subsequent mistakes with line numbers followed through) or additional incorrect output being given.

Candidates should be encouraged to simply leave boxes blank if no output is given on a particular line. If (for example) 'x' is written, examiners are unsure whether the candidate meant no output or the letter 'x' should be output. This ambiguity would mean that no mark could be given.

Question 9 (c) (i)

(c) The height a student jumps in the high jump needs to be input and validated.
The height is entered in centimetres (cm) and must be between 40.0 and 180.0 inclusive.

(i) Write an algorithm to:
- take the height jumped as input
- output "VALID" or "NOT VALID" depending on the height input.

You must use **either**:
- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied.

.....

.....

.....

.....

.....

..... **[4]**

This question was done very well by the majority of candidates, with multiple ways of achieving the marks possible.

One common problem was consistent with Question 3 (b), as again multiple conditions could be evaluated using a single `if` statement. Candidates needed to refer to their input value for each comparison if they were to achieve full marks. Another common problem was with the boundaries used; the tests must check 40 to 80 inclusive (for valid response) to be marked as correct. Obviously, if an input on these boundaries would produce the wrong output then full marks could not be given.

One final common problem involved the use of greater than or equal to signs (and also less than or equal to). The common signs used in mathematics are not available on a typical keyboard and so would not be allowed in Section B of this paper. Instead, **>=** or **<=** should be used, and these should be the correct way around.

Question 9 (d)

(d) The individual results for each student in each event are stored in a database.

The database table `TblResult` stores the times of students in the 100 m race. Some of the data is shown:

StudentID	YearGroup	TeamName	Time
11GC1	11	Valiants	20.3
10VE1	10	Super-Team	19.7
10SM1	10	Super-Team	19.2
11JP2	11	Champions	19.65

Complete the SQL statement to show the Student ID and team name of all students who are in year group 11

```
SELECT StudentID, .....

FROM .....

.....
```

[4]

A number of candidates struggled with this question. Problems included spaces in field names, misspelling of the table name (such as `TblResult`**`s`** plural when `TblResult` singular was given) or misunderstanding of the WHERE clause.

Allowance was given where == was used for comparison and examiners were instructed to allow this (as this is used for comparison in high-level languages such as Python), although this is incorrect as defined in the most recent ANSI SQL standards.

Question 9 (e) (i) and e (ii)

(e) Abstraction and decomposition have been used in the design of the sports day program.

(i) Identify **one** way that abstraction has been used in the design of this program.

.....

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

(ii) Identify **one** way that decomposition has been used in the design of this program.

.....

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

Both questions here asked about abstraction and decomposition of the sports day program. As explained previously in this report, where a scenario or context is given, candidates are expected to use this context. No marks were given by examiners for generic definitions of what the term abstraction or decomposition means.

Abstraction in the sports day program could have been for focusing on anything sensible (such as event names) or removing/ignoring anything sensible (such as showing student IDs instead of names). Where the context of the sports day was used, candidates were generally successful in achieving this mark.

Decomposition use was more tricky to correctly identify, as many candidates simply referred to how already separate data was stored. Where this extended to true decomposition (such as breaking down data into multiple tables, splitting up event data, etc.) this was credited but the average candidate fell short here. A much more successful approach was to discuss the decomposition of the program, such as having a separate algorithm for each event. Candidates attempting this angle of response did very well.

Examiners were instructed for both questions to be generous in deciding whether candidates had indeed referred to the sports day context.

Question 9 (f)

(f) An algorithm works out which team has won (has the highest score).

Write an algorithm to:
- prompt the user to enter a team name and score, or to enter "stop" to stop entering new teams
- repeatedly take team names and scores as input until the user enters "stop"
- calculate which team has the highest score
- output the team name and score of the winning team in an appropriate message.

You must use **either**:
- OCR Exam Reference Language, **or**
- A high-level programming language that you have studied

.....

.....

.....

.....

.....

.....

[6]

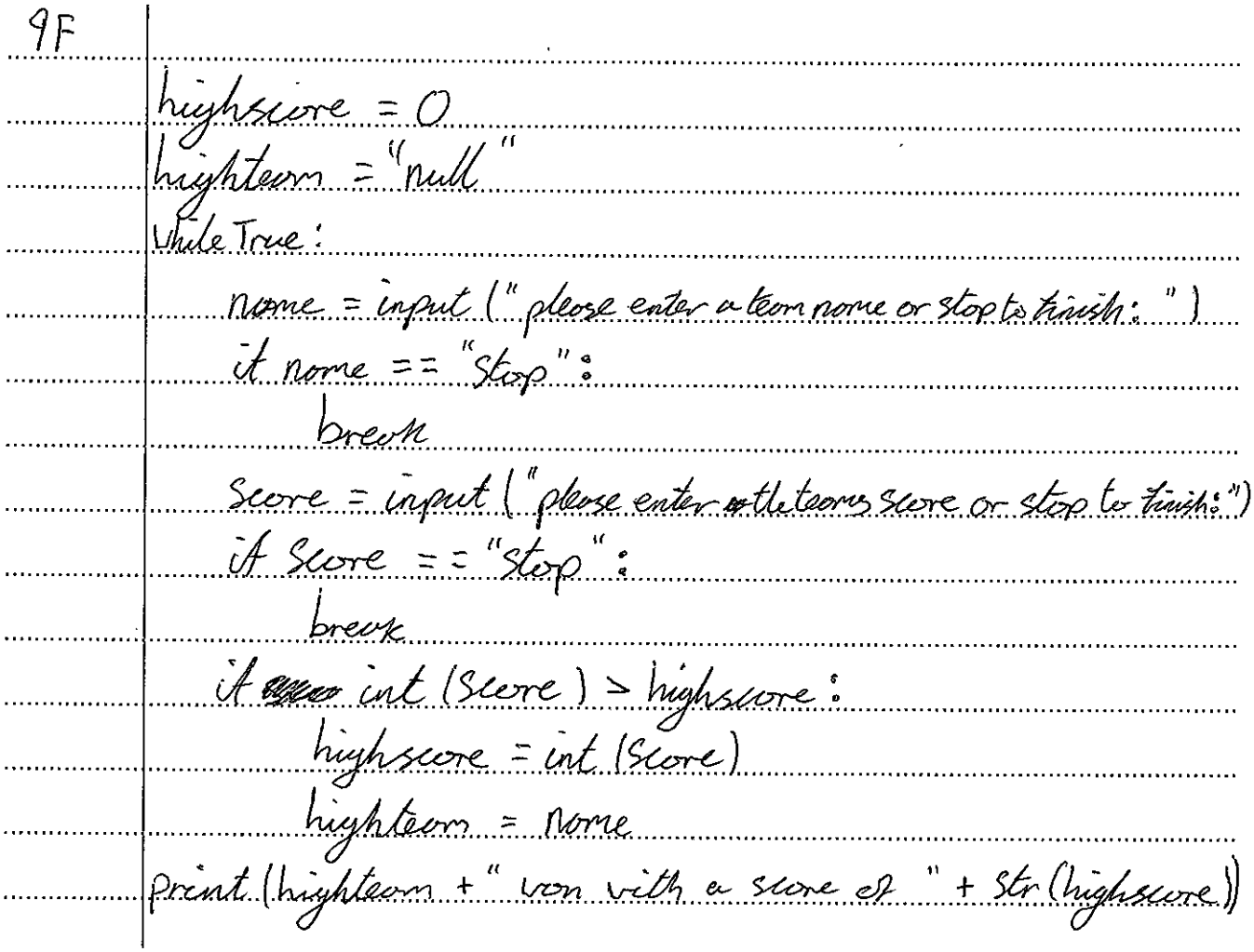
The final question in Section B is expected to be challenging and this proved to be the case, although again was perhaps more accessible than previous papers' final questions.

Marks were available for inputs (one mark) and correctly iterating over as required (two marks), with these three marks proving to be the easiest to achieve. The next three marks required significant processing in terms of calculating the highest score and team name from multiple values entered by the user. The vast majority of candidates simply kept a running 'highest score' and updated this on each iteration. Where this was attempted, it was mostly successful. Other candidates attempted more complex solutions, including adding data to arrays and then calculating highest values; where this was done successfully, this of course achieved full marks but frequently small logic mistakes meant that not all marks were given. Centres should encourage candidates to keep their responses simple and not produce over-elaborate solutions if a simpler alternative is available.

A common mistake was where candidates attempted to use loops in the style of `for x in list:`. In this case, the variable `x` is a reference to an item in the array and not an index. It would therefore not be appropriate to try to access `list[x]` later in the code.

A significant number of candidates were able to create a solution that fully met the requirements of the question, doing so in an elegant and efficient manner. This is extremely pleasing and show excellent understanding, produced from excellent teaching and significant amounts of practical programming experience.

Exemplar 3



The candidate response shown here achieved six out of six marks. Both the `name` and `score` are input as required, with this being inside a while loop. Perhaps unconventionally (but acceptably), the candidate has used a `break` command to end the loop (which otherwise is infinite) upon stop being entered. This could have been more elegantly rewritten as `while name != 'stop'` but this would not have achieved any further marks.

Within the loop, the candidate uses two variables to keep track of the current highest score and associated team, before printing these out in a message once the loop has ended.

Supporting you

Teach Cambridge

Make sure you visit our secure website **Teach Cambridge** to find the full range of resources and support for the subjects you teach. This includes secure materials such as set assignments and exemplars, online and on-demand training.

Don't have access? If your school or college teaches any OCR qualifications, please contact your exams officer. You can forward them this link to help get you started.

Reviews of marking

If any of your students' results are not as expected, you may wish to consider one of our post-results services. For full information about the options available visit the OCR website.

Access to Scripts

We've made it easier for Exams Officers to download copies of your candidates' completed papers or 'scripts'. Your centre can use these scripts to decide whether to request a review of marking and to support teaching and learning.

Our free, on-demand service, Access to Scripts is available via our single sign-on service, My Cambridge. Step-by-step instructions are on our website.

Keep up-to-date

We send a monthly bulletin to tell you about important updates. You can also sign up for your subject specific updates. If you haven't already, sign up here.

OCR Professional Development

Attend one of our popular professional development courses to hear directly from a senior assessor or drop in to a Q&A session. Most of our courses are delivered live via an online platform, so you can attend from any location.

Please find details for all our courses for your subject on **Teach Cambridge**. You'll also find links to our online courses on NEA marking and support.

Signed up for ExamBuilder?

ExamBuilder is a free test-building platform, providing unlimited users exclusively for staff at OCR centres with an Interchange account.

Choose from a large bank of questions to build personalised tests and custom mark schemes, with the option to add custom cover pages to simulate real examinations. You can also edit and download complete past papers.

Find out more.

Active Results

Review students' exam performance with our free online results analysis tool. It is available for all GCSEs, AS and A Levels and Cambridge Nationals (examined units only).

Find out more.

You will need an Interchange account to access our digital products. If you do not have an Interchange account please contact your centre administrator (usually the Exams Officer) to request

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
facebook.com/ocrexams
twitter.com/ocrexams
instagram.com/ocrexaminations
linkedin.com/company/ocr
youtube.com/ocrexams

We really value your feedback

Click to send us an autogenerated email about this resource. Add comments if you want to. Let us know how we can improve this resource or what else you need. Your email address will not be used or shared for any marketing purposes.

I like this **I dislike this**

Please note – web links are correct at date of publication but other websites may change over time. If you have any problems with a link you may want to navigate to that organisation's website for a direct search.

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.

You can copy and distribute this resource in your centre, in line with any specific restrictions detailed in the resource. Resources intended for teacher use should not be shared with students. Resources should not be published on social media platforms or other websites.

OCR acknowledges the use of the following content: N/A

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.

Question 3 (a)

3

(a) Show the steps that a merge sort would take to put the data set into ascending numerical order.

2, 7, 3, 1, 9, 6, 5, 4

[4]

The majority of candidates showed clearly that they understood the steps a merge sort takes to sort values into ascending order. The strongest responses clearly showed both the divide stage (splitting values up into individual lists) and then the conquer stage (merging lists together in order).

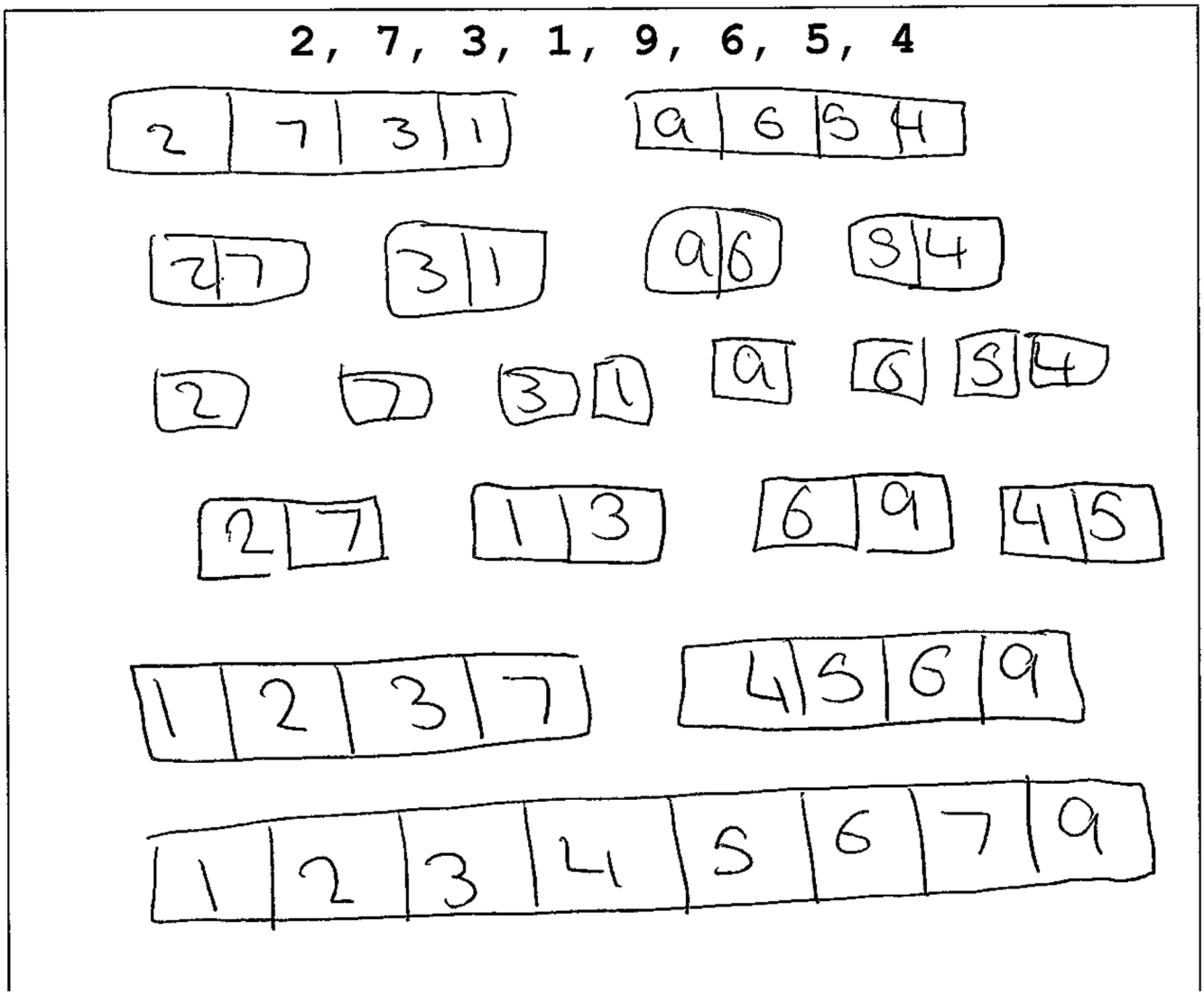
Only a minority of candidates attempted a different sorting algorithm or showed a lack of understanding of the abstract idea behind the algorithm; this is very pleasing to see.

However, several candidates still merge lists together into an unsorted list and then take a second step to sort the list. This has been commented on in previous Examiner's Reports but remains a common misconception. Where this was the case in candidate answers, the mark scheme instructed examiners not to give the relevant mark.

Misconception

A merge sort merges smaller lists together to form a larger sorted list. At each step of the merge process, **all** lists are **always** in order; it is the process of merging together that performs the sort. A very common misconception is for candidates to merge lists together by simply appending one to the other (to form an unsorted list) and then sorting this list. Of course, this begs the recursive question - what algorithm is being used to carry out the sort?

Exemplar 1



In this example, the candidate has successfully achieved full marks. The list is repeatedly divided into smaller and smaller lists until lists of size one are present. These are then merged together into lists of size 2, then lists of size 4 before finally being merged into one final list of size 8.

Note that it is the act of merging that does the sorting; values are **not** merged out of order and then sorted.

The candidate puts boxes around their numbers to clearly indicate the different lists. This is an example of best practice.

Question 3 (b)

(b) Tick (✓) **one** box in each table to complete the descriptions.

An insertion sort…

	Tick (✓) **one** box
uses an array of numbers with a sorted part and unsorted part	
picks out the middle value to begin with	
merges arrays of numbers together	

A bubble sort compares pairs of values and then:

	Tick (✓) **one** box
always swaps them	
swaps them if they are in the correct order	
swaps them if they are in the incorrect order	

A sorting algorithm can sort data in an array. An array:

	Tick (✓) **one** box
is declared with a fixed length	
stores multiple values that cannot be changed while the program is running	
can only be declared with a maximum of one-dimension (1D)	

[3]

The majority of candidates were able to select the correct answers for the first two tick boxes, showing good understanding of sorting algorithms. Fewer candidates were able to define an array as being declared with a fixed length. This may be due to Python being the most common programming language used, and Python support **lists** by default instead of arrays.

Lists are dynamic data structures and so can be appended to, whereas arrays are static data structures with a fixed length.

Assessment for learning

If Python is used as the main programming language in a centre, care must be taken to make sure that elements of the specification not easily covered in Python, (e.g. arrays, `do...until` loops, `switch/case`) are covered suitably in a different way – for example, perhaps by introducing a second programming language.

Note : `Switch/case` is supported in Python version 3.10 and later using the `match` and `case` keywords.

Question 3 (c) (i) and (ii)

(c) An algorithm can be used to search for a value in an array. The algorithm follows these steps:

- compares the first value in the array to the value being searched for
- stops if the comparison is `true`
- moves to the next value in the array if the comparison if `false`.

(i) Give the name of the searching algorithm described.

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

(ii) Identify **one other** condition that will cause the algorithm to stop.

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

Most candidates were able to identify the algorithm given as a linear search.

The other stopping condition for part (ii) had to be precisely and correctly defined by candidates. Many responded “when the value is not found”, but how would this be written as a condition for a high-level programming language to check? The condition is technically when the end of the list is reached **and** the value has not been found, but responses relating to simply reaching the end of the list or having checked each item were accepted.