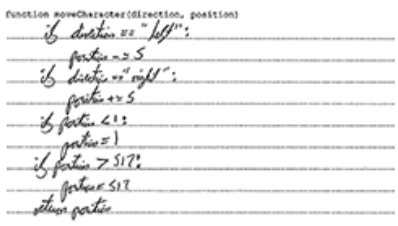


Mark scheme

Question			Answer/Indicative content	Marks	Guidance
1			<p>1 mark each to max 6</p> <ul style="list-style-type: none"> 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 <code>position</code> (or equivalent) ...correctly checking if direction is "right" and adding 5 to <code>position</code> (or equivalent) Ensuring position (or equivalent) is between 1 and 512 inclusive Returning the updated position <p><u>Example</u></p> <pre>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</pre>	6 (AO 3)	<p>Allow <code>else</code> for BP3/4 (validated in question 8a)</p> <p>Allow <code><=</code>, <code>>=</code> and equivalents (e.g. <code><= 0</code>) for BP5.</p> <p>Do not award BP5 if before BP3 and 4 (otherwise will alter position value)</p> <p>BP6 only to be given if attempt made at calculating new position. Calculation can be partial/incorrect.</p> <p>Ignore repeat of function header / end.</p> <p>Accept flowchart / structured English but must not just repeat the question.</p> <p>If response uses loop to incorrectly change position multiple times, do not award BP1 (incorrect overwrite)</p> <p>For minor syntax errors (e.g. missing quotation marks or <code>==</code> for assignment, spaces in variable names) penalise once then FT.</p> <p><u>Examiner's Comments</u></p> <p>This question was an excellent discriminator in terms of candidate achievement. In particular, correct use of the given parameters (<code>direction</code> and <code>position</code>) was only seen from the most successful candidates. Many candidates instead started their response by asking for input from the user and therefore overwriting the parameters, losing crucial marks in the process.</p>

					<p>As on previous papers, this question provided one mark for any attempt made at a section of the requirement, in this case selection. Candidates who attempted to use an <code>if</code> or <code>select case</code> statement, even incorrectly or in the wrong context were able to achieve at least this mark; centres should therefore continue to encourage all candidates to attempt each and every question and not leave responses blank.</p> <p>It was challenging to achieve full marks, but many candidates did because of their excellent levels of practical programming experience in school.</p> <p>Exemplar 2</p>  <pre> function moveCharacter(direction, position) if direction == "left" position -= 5 if direction == "right" position += 5 if position < 1 position = 1 if position > 512 position = 512 return position </pre> <p>This candidate achieved full marks. The given parameters (direction and position) are both used and not overwritten by inputs before the position is modified depending on the direction given. The position is then validated to make sure that it is between 1 and 512 before being returned.</p> <p>Note the use of <code>position += 5</code> to add 5 to the position. This is an entirely acceptable alternative to <code>position = position + 5</code></p>
			Total	6	
2	a	i	<ul style="list-style-type: none"> String Integer Real / Float 	3 (AO 3)	<p>Accept alternative equivalent correct data types (e.g. single/double/decimal for BP3)</p> <p>Do not accept char for BP1</p>

					<p><u>Examiner's Comments</u></p> <p>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.</p>
		ii	<ul style="list-style-type: none"> • <code>theTeam.length() - 1 / 5</code> • <code>count</code> • <code>studentName</code> • <code>True</code> 	<p>4 (AO 3)</p>	<p>Accept <code>6 / theTeam.length()</code> for BP1 (Python).</p> <p>Accept alternative length functions e.g. <code>len()</code></p> <p>Accept <code>count = 5</code> (and equivalents) for BP1. Accept "True" for BP4.</p> <p>Do not allow obvious spaces in variable names.</p> <p>Ignore capitalisation.</p> <p><u>Examiner's Comments</u></p> <p>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 <code>True</code> 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.</p> <p>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</p>

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.

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

Example

Line number	javelinThrow	yearGroup	score	Output
01	14.3			
02		10		
06			2	
11			4	
13				The score is 4

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.

Max 3 if in wrong order or additional (incorrect) changes. Penalise line numbers once then FT.

Allow FT for BP4 for current value of `score`.

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".

Examiner's Comments

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

4
(AO
3)

b

				<p>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.</p> <p>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.</p>
	c i	<ul style="list-style-type: none"> inputs a value from the user <u>and stores/uses</u> checks min value ($\geq 40.0 / < 40$) checks max value ($\leq 180.0 / > 180$) ...outputs both valid / not valid correctly <u>based on checks</u> <p><u>Example 1 (checking for valid input)</u></p> <pre>h = input("Enter height jumped") if h >= 40 and h <= 180 then print("valid") else print("not valid") endif</pre> <p><u>Example 2 (checking for invalid input)</u></p> <pre>h = input("Enter height jumped") if h < 40 or h > 180 then print("not valid") else print("valid") endif</pre>	<p>4 (AO 3)</p>	<p>Answers using AND/OR for BP2 and BP3 must be logically correct e.g. if height ≥ 40 and <u>height</u> ≤ 180. Do not accept if height ≥ 40 and ≤ 180</p> <p>Answers using OR will reverse output for BP4 (see examples).</p> <p>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.</p> <p>Allow FT for BP4 if reasonable attempt at validating (must include at least one boundary)</p> <p>Ignore conversion to int on input. <code>input</code> cannot be used as a variable name.</p> <p>Greater than / less than symbols must be appropriate for a high-level language / ERL. Do not accept \Rightarrow (wrong way around) or \geq (not available on keyboard). No obvious spaces in variable names. Penalise once and then FT.</p>

				<p><u>Examiner's Comments</u></p> <p>This question was done very well by the majority of candidates, with multiple ways of achieving the marks possible.</p> <p>One common problem was consistent with Question 3 (b), as again multiple conditions could be evaluated using a single <code>if</code> 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.</p> <p>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, <code>>=</code> or <code><=</code> should be used, and these should be the correct way around.</p>
		ii	<ul style="list-style-type: none"> Any normal value (between 40 and 180 inclusive) 40.0 / 180.0 Any value less than 40 / any value greater than 180 / any non-numeric value 	<p>3 (AO 3)</p> <p>No need to include decimals, e.g. accept 50. Ignore cm if given.</p> <p>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</p>
		d	<ul style="list-style-type: none"> <code>TeamName</code> only in first space <code>TblResult</code> in second space <code>WHERE</code> <code>...YearGroup = 11</code> 	<p>4 (AO 3)</p> <p>Max 3 if not in correct order / includes other logical errors.</p> <p>Ignore capitals. Do not accept * or additional fields for BP1</p> <p>Spelling must be accurate (e.g. not</p>

				<p>TblResults).</p> <p>No spaces in field names, penalise obvious spaces once and then FT. Allow quotation marks around field names, table name and 11</p> <p>Accept == for BP4 (invalid SQL but works in some environments)</p> <p><u>Examiner's Comments</u></p> <p>A number of candidates struggled with this question. Problems included spaces in field names, misspelling of the table name (such as TblResults plural when TblResult singular was given) or misunderstanding of the WHERE clause.</p> <p>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.</p>
	e i	<ul style="list-style-type: none"> any example of simplification / removing data or focussing on data (in the design) <p><u>Examples :</u></p> <ul style="list-style-type: none"> - "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 (AO 3)	<p>Must be applicable to <u>this program</u> (in the context of students and a sports day), not a generic description of what abstraction is. Give BOD where this is unclear.</p> <p><u>Examiner's Comments</u></p> <p>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.</p> <p>Abstraction in the sports day program could have been for</p>

				<p>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.</p> <p>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.</p> <p>Examiners were instructed for both questions to be generous in deciding whether candidates had indeed referred to the sports day context.</p>
	ii	<ul style="list-style-type: none"> any example of breaking down the program into sections/subroutines any example of breaking down the database into tables <p><u>Examples :</u></p> <ul style="list-style-type: none"> - “splits the program up into different events” - “separates the validation routines into subroutines” - “breaks the database down into a table per event” 	<p>1 (AO 3)</p>	<p>Must be applicable to <u>this program</u>, not a generic description of what decomposition is. Give BOD where this is unclear.</p> <p>Do not give answers discussing splitting into fields (e.g. split into StudentID, YearGroup, etc).</p> <p>BOD if answer discusses one table but suggests other tables could be used.</p> <p>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.</p> <p>Allow reference to sports day to</p>

				<p>mean sports day program.</p> <p><u>Examiner's Comments</u></p> <p>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.</p> <p>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.</p> <p>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.</p> <p>Examiners were instructed for both questions to be generous in deciding whether candidates had indeed referred to the sports day context.</p>
	f		<ul style="list-style-type: none"> Input team name AND score and store / use separately Attempt at using iteration... 	<p>6 (AO 3)</p> <p>For BP3, allow "stop" to be entered for either team name or score (or both). Allow third input (e.g. "do you</p>

		<ul style="list-style-type: none"> • ...to enter team/score until "stop" entered • Calculates highest score • Calculates winning team name... • ...Outputs highest score and team name <p>Example 1</p> <pre>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)</pre> <p>Example 2 (alternative)</p> <pre>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)] print(highscore) print(highteam)</pre>	<p>wish to stop?")</p> <p>Allow use of <code>break</code> (or equivalent) to exit loop for BP3.</p> <p>Allow use of recursive function(s) for BP2/3.</p> <p>Initialisation of variables not needed - assume variables are 0 or empty string if not set.</p> <p>Ignore that multiple teams could get the same high score, assume only one team has the highest score.</p> <p>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.</p> <p>FT for BP6 if attempt made at calculating highest score/name</p> <p>If answer simply asks for multiple entries (not using iteration), BP2 and 3 cannot be accessed but all others available.</p> <p>For minor syntax errors (e.g. missing quotation marks or == for assignment, spaces in variable names) penalise once then FT.</p> <p><code>input</code> cannot be used as a variable name.</p> <p><u>Examiner's Comments</u></p> <p>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.</p> <p>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</p>
--	--	--	---

				<p>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.</p> <p>A common mistake was where candidates attempted to use loops in the style of <code>for x in list :</code>. In this case, the variable <code>x</code> is a reference to an item in the array and not an index. It would therefore not be appropriate to try to access <code>list[x]</code> later in the code.</p> <p>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.</p> <p>Exemplar 3</p>
--	--	--	--	--

				<pre>9F highscore = 0 highteam = "null" while True: name = input("Please enter a team name or stop to finish: ") if name == "stop": break score = input("Please enter a team score or stop to finish: ") if score == "stop": break if int(score) > highscore: highscore = int(score) highteam = name print(highteam + " won with a score of " + str(highscore))</pre>	<p>The candidate response shown here achieved six out of six marks. Both the <code>name</code> and <code>score</code> are input as required, with this being inside a while loop. Perhaps unconventionally (but acceptably), the candidate has used a <code>break</code> command to end the loop (which otherwise is infinite) upon stop being entered. This could have been more elegantly rewritten as <code>while name != 'stop'</code> but this would not have achieved any further marks.</p> <p>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.</p>														
			Total	30															
3			<table><tr><th rowspan="2">Keyword</th><th colspan="2">Programming construct</th></tr><tr><th>Selection</th><th>Iteration</th></tr><tr><td>if</td><td>✓</td><td></td></tr><tr><td>for</td><td></td><td>✓</td></tr><tr><td>while</td><td></td><td>✓</td></tr></table>	Keyword	Programming construct		Selection	Iteration	if	✓		for		✓	while		✓	3 (AO 1)	<p><u>Examiner's Comments</u></p> <p>This question was answered very well by the majority of candidates, showing a good understanding of the difference between the key programming constructs of selection and iteration.</p> <p>Practical programming skills in lessons should be used to make this explicit link between technical definitions and use within a high-level language.</p>
Keyword	Programming construct																		
	Selection	Iteration																	
if	✓																		
for		✓																	
while		✓																	
			Total	3															

4	a	<p>1 mark for each output</p> <table><tr><td><pre>print(message.upper)</pre></td><td>ABCD1234 (upper case)</td></tr><tr><td><pre>print(message.left(4))</pre></td><td>abcd (lower case)</td></tr><tr><td><pre>print(int(message.right(4))*2)</pre></td><td>2468</td></tr></table>	<pre>print(message.upper)</pre>	ABCD1234 (upper case)	<pre>print(message.left(4))</pre>	abcd (lower case)	<pre>print(int(message.right(4))*2)</pre>	2468	<p>Case must be correct but BOD if ambiguous.</p> <p>Allow quotation marks in answer.</p> <p><u>Examiner's Comments</u></p> <p>This question assessed string manipulation and used OCR Exam Reference Language(ERL) to present each statement. This highlights the importance of candidates being taught how to interpret ERL as questions in the exam will be written in this format.</p> <p>Candidates were generally confident with the use of .upper and .left(). Candidates were less confident when this was combined with casting in the last part.</p> <p>3 (AO 2)</p>
<pre>print(message.upper)</pre>	ABCD1234 (upper case)								
<pre>print(message.left(4))</pre>	abcd (lower case)								
<pre>print(int(message.right(4))*2)</pre>	2468								
	b	<p>1 mark per bullet point :</p> <ul style="list-style-type: none">• storing both strings correctly in word1 and word2• correct concatenation (word1 then word2)...• ...storing in variable <u>message</u> <p><u>Example</u></p> <pre>word1 = "Hello" word2 = "Everyone" message = word1 + word2</pre>	<p>Accept & / + / . etc as valid methods of concatenation. Allow use of sensible concatenation functions e.g. concat() . Do not allow commas.</p> <p>Do not allow == for assigning value to string. Do not allow spaces in variable names. Penalise once then FT.</p> <p>Ignore additional code given. Ignore case.</p> <p>Reasonable attempt at BP2 needed to access BP3.</p> <p><u>Examiner's Comments</u></p> <p>This question was intended to be straightforward and aimed at all candidates, including those expecting to achieve lower grades. However, a significant number of candidates dropped marks, either through perhaps not reading the question requirements or through not understanding the term concatenation.</p> <p>3 (AO 3)</p>						

					<p>There is no requirement in the question to print out the message obtained, simply to store it in the variable <code>message</code>. Where this was done in addition, this was not penalised but often it was done instead of storing the concatenated message.</p> <p>A common mistake was to use the comma for concatenation when in most high-level languages (and especially Python), this is simply used to print out two items or pass two arguments to a subroutine and not actually concatenate values. Another common mistake was with the names of the variables given – if these included spaces or differed from those given in the question (such as <code>word 1</code>, <code>wordone</code> or <code>world1</code> instead of <code>word1</code>) examiners were instructed to not allow these alternatives.</p>
			Total	6	
5	a		7	1 (AO 3 1)	Correct Answer Only
	b		4	1 (AO 3 1)	Correct Answer Only
	c		6	1 (AO 3 1)	Correct Answer Only
	d		6	1 (AO 3 1)	Correct Answer Only
			Total	4	
6	a		<p>One mark per bullet point</p> <ul style="list-style-type: none"> Ask the user for two inputs and store/use these Open <code>pilots.txt</code> (for write/append) Write both inputs to opened text file 	4 (AO 3 2b)	<p>Award BP4 for implicit closing of file (e.g. using <code>with...</code> in Python)</p> <p>Example answer :</p> <pre>dronepilotData = open("pilots.txt") pilotCode = input("enter code")</pre>


			<ul style="list-style-type: none">Close text file		<pre>dob = input("enter date of birth") dronepilotData.writeLine(pilotCode, dob) dronepilotData.close()</pre> <p>Allow data to be written either by simply writing both values or by concatenating into one string with a separating comma.</p>						
	b		<p>One mark per bullet point to max 6</p> <ul style="list-style-type: none">Function header <code>pilotValid()</code> with (at least one) parameterChecks array element <code>[i,0]</code> for each item......calculates total...casting to float / realChecks if 9 hours or fewerReturns a value...returns "warning" and "valid" correctly	6 (AO 3 2b)	<p>Example answer :</p> <pre>function pilotValid(pilotCode) total = 0 status = "" for i = 0 to 5 if journeys[i,0] == pilotCode then temp = float(journeys[i,1]) total = total + temp endif next i if total > 9 then status = "warning" else status = "valid" endif return status endfunction</pre> <p>Do not allow casting to integer for BP4, data shows some journeys have decimal places.</p> <p>Allow FT for BP5 if attempt made at calculation.</p> <p>Allow FT for BP7 if value is output instead of returned.</p>						
			Total	10							
7			<p>1 mark per row</p> <table><tr><th>Operator</th><th>Comparison</th><th>Arithmetic</th></tr><tr><td>==</td><td>✓</td><td></td></tr></table>	Operator	Comparison	Arithmetic	==	✓		4 (AO 1 1a)	
Operator	Comparison	Arithmetic									
==	✓										

			<table><tr><td>+</td><td></td><td>✓</td></tr><tr><td>DIV</td><td></td><td>✓</td></tr><tr><td>></td><td>✓</td><td></td></tr></table>	+		✓	DIV		✓	>	✓			
+		✓												
DIV		✓												
>	✓													
			Total	4										
8	a		<ul style="list-style-type: none">number with decimal places / fractional part	1 (AO 1 1a)	Do not accept examples on their own.									
	b	i	<ul style="list-style-type: none">Boolean has two possible values / True or False / Result has three possible values / result has more than two options	1 (AO 2 1a)	The results are words and not Boolean values.									
		ii	1 mark for data type, 1 mark for matching justification. <ul style="list-style-type: none">String......can store "win", "loss" or "draw" / equivalentChar......can store "W", "L" or "D" / equivalentInteger / Real......can store 0, 1, 2 / equivalent	2 (AO 2 1a)	Allow other sensible explanation that shows candidate has considered how each of the three states could be stored.									
	c	i	1 mark per bullet point <ul style="list-style-type: none">Input code from user <u>and store/use</u>Repeat for non-three character codesCheck for one code......and set level appropriatelyCheck for and set level for 2nd codeSet level to 1 for any other 3 character code	6 (AO 3 2b)	Example answer : level = 0 code = "" while code.length != 3 then code = input("enter a 3 character code") endwhile switch code : case "SVA": level = 2 case "UTV": level = 3 default: level = 1 endswitch alternative example answer level = 0 code = ""									

					<pre> while code.length != 3 then code = input("enter a 3 character code") endwhile if code == "SVA" then level = 2 elseif code == "UTV" then level = 3 else level = 1 endif </pre>
		ii	<p>1 mark per bullet point</p> <ul style="list-style-type: none"> function <code>nextlevel()</code> defined with <u>at least</u> one parameter returns parameter plus 1... ... except for if parameter is 3, then returns 1 	<p>3 (AO 3 2b)</p>	<p>Example answer :</p> <pre> function nextlevel(oldlevel) if oldlevel == 3 then return 1 else return oldlevel + 1 endif end function </pre> <p>Accept correct use of MOD for BP2 and 3.</p> <p>Allow FT for BP3 if value output / calculated but not returned.</p>
		iii	<ul style="list-style-type: none"> <code>print(nextlevel(3))</code> 	<p>1 (AO 3 2b)</p>	<p>Allow other logically correct answers</p>
			Total	14	
9	a	i	<ul style="list-style-type: none"> DroneID, Mileage FROM AND > 50000 	<p>4 (AO 3 2b)</p>	<p>Accept SELECT * or selection of additional fields for BP1.</p>
		ii	<ul style="list-style-type: none"> if <u>Mileage-LastCheck > 10000</u> Output "Check" and "No Check" or equivalent correctly <u>based on logical check for BP1</u> 	<p>2 (AO 3 2a)</p>	<p>BP1 could be >, < or either in words.</p> <p>Ignore case and minor misspellings.</p> <p>BP2 (output) could be either way around depending on comparison for BP1.</p>

	b		<p>1 mark per row</p> <ul style="list-style-type: none">c = 90 on line 05c = 900 on line 05pilotCode = HK900 on line 07HK900 output on line 08	4 (AO 3 1)	<p>Ignore additional lines that do not affect outcome. FT for missing or incorrect line numbers. FT for output based on incorrect tracing of loop.</p> <table><tr><th>Line number</th><th>a</th><th>b</th><th>c</th><th>pilotCode</th><th>Output</th></tr><tr><td>01</td><td>H</td><td></td><td></td><td></td><td></td></tr><tr><td>02</td><td></td><td>K</td><td></td><td></td><td></td></tr><tr><td>03</td><td></td><td></td><td>9</td><td></td><td></td></tr><tr><td>05</td><td></td><td></td><td>90</td><td></td><td></td></tr><tr><td>05</td><td></td><td></td><td>900</td><td></td><td></td></tr><tr><td>07</td><td></td><td></td><td></td><td>HK900</td><td></td></tr><tr><td>08</td><td></td><td></td><td></td><td></td><td>HK900</td></tr></table>	Line number	a	b	c	pilotCode	Output	01	H					02		K				03			9			05			90			05			900			07				HK900		08					HK900
Line number	a	b	c	pilotCode	Output																																																
01	H																																																				
02		K																																																			
03			9																																																		
05			90																																																		
05			900																																																		
07				HK900																																																	
08					HK900																																																
	c		<p>One mark per bullet point</p> <ul style="list-style-type: none">Any value between 0 and 20 (e.g. 4)TrueInvalid / erroneous / sensible alternativeFalse	4 (AO 3 2c)	<table><tr><th>Experience in years</th><th>Type of test</th><th>Expected output</th></tr><tr><td>4</td><td>Normal</td><td>True</td></tr><tr><td>20</td><td>Boundary</td><td>True</td></tr><tr><td>32</td><td>Erroneous/Invalid</td><td>False</td></tr></table>	Experience in years	Type of test	Expected output	4	Normal	True	20	Boundary	True	32	Erroneous/Invalid	False																																				
Experience in years	Type of test	Expected output																																																			
4	Normal	True																																																			
20	Boundary	True																																																			
32	Erroneous/Invalid	False																																																			
	d i		<ul style="list-style-type: none">Attempt at using selectionCalculates pay correctly for pilots with less than 2 years experienceCalculates pay correctly for pilots with 2 to 5 years experience. Must include both 2 and 5.Calculates pay correctly for pilots with more than 5 years experience	4 (AO 3 2b)	<p>Example answer :</p> <pre>experience = input("Enter years of experience") miles = input("Enter miles flown") totalPay = 0 if exp <2 then totalPay = 120+(0.45*miles) elseif exp <=5 then totalPay = 150+(0.65*miles) else</pre>																																																

					<pre>totalPay = 180+(0.85*miles) end if print(totalPay)</pre> <p>FT for BP4 only where a reasonable attempt at calculating pay has been made.</p>
		ii	<ul style="list-style-type: none"> totalPay (experience, miles) / (miles, experience) 	2 (AO 3 2c)	<pre>totalPay = calculatePay(experience, miles)</pre>
			Total	20	
1 0			<p>1 mark</p> <ul style="list-style-type: none"> 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 <p>1 mark</p> <ul style="list-style-type: none"> 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 More efficient than / does not need to iterate as many times as count controlled / for loop 	2 (AO 2 1b)	<p>Max 1 from each section, 2 marks total.</p> <p>Do not allow “while names are in the wrong order”.</p> <p>BP4 must have reference to <u>checking</u> a condition / condition being met, not just having a condition.</p> <p><u>Examiner’s Comments</u></p> <p>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.</p> <p>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.</p> <p>Candidates generally found this question challenging. Many</p>

						<p>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 <code>name</code> 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 <code>name</code> is in the correct position (or even better, an explanation of how this is decided on) which ends the loop.</p> <div> Assessment for learning</div> <p>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.</p>																													
			Total	2																															
1 1	a		<p>1 mark for each row</p> <table><tr><th>Variable</th><th>Boole an</th><th>Ch ar</th><th>Stri ng</th><th>Integ er</th><th>Re al</th></tr><tr><td>UserName</td><td></td><td></td><td>✓</td><td></td><td></td></tr><tr><td>EmergencyPhone Number</td><td></td><td></td><td>✓</td><td></td><td></td></tr><tr><td>DoorSensorActi ve</td><td>✓</td><td></td><td></td><td></td><td></td></tr><tr><td>DoorActiveTime</td><td></td><td></td><td></td><td>✓</td><td></td></tr></table>	Variable	Boole an	Ch ar	Stri ng	Integ er	Re al	UserName			✓			EmergencyPhone Number			✓			DoorSensorActi ve	✓					DoorActiveTime				✓		4 (AO 3 2a)	<p>No mark if more than 1 tick on a row.</p> <p>Allow other indications of choice (e.g. cross) as long as clear.</p>
Variable	Boole an	Ch ar	Stri ng	Integ er	Re al																														
UserName			✓																																
EmergencyPhone Number			✓																																
DoorSensorActi ve	✓																																		
DoorActiveTime				✓																															
	b		<p>1 mark each:</p> <ul style="list-style-type: none">Attempt at using selection / condition controlled loop	4 (AO 3 2b)	<p>Selection could be done using IF statement, case statement or any other sensible valid method.</p>																														

		<ul style="list-style-type: none"> • Checking if system armed / while system armed • If Door Sensor active OR Window Sensor active (both checks required) • calling SoundAlarm correctly 	<p>Allow reference to AlarmActivated or equivalent instead of SystemArmed</p> <p>Ignore any inputs or modification of variables.</p> <p>Allow True / False as strings. Allow checking against strings (e.g. if SystemArmed == "active")</p> <p>Allow checking armed/disarmed for BP2 and BP3</p> <p>Only award BP4 if SoundAlarm correctly called / not called in every situation. If issues on previous lines (e.g. lack of brackets where needed) means this is not the case, do not award BP4.</p> <p>Checking could be done by evaluating variable directly (if SystemArmed) or by comparison (if SystemArmed == True)</p> <p><u>Example answer 1</u></p> <pre>if SystemArmed then if DoorSensorActive then SoundAlarm() else if WindowSensorActive then SoundAlarm() endif endif</pre> <p><u>Example answer 2</u></p> <pre>while SystemArmed then if DoorSensorActive then SoundAlarm() else if WindowSensorActive then SoundAlarm() endif endif</pre> <p><u>Example answer 3</u></p>
--	--	---	--

```

if SystemArmed and
(DoorSensorArmed or
WindowSensor) then
SoundAlarm()
endif

```

Note – above example needs brackets,

```

if SystemArmed and
DoorSensorArmed or
WindowSensor then

```

is not logically valid for this scenario
(will sound alarm when not armed if window sensor is active)

Example answer 4

```

if SystemArmed and
DoorSensorArmed
SoundAlarm()
else if SystemArmed and
WindowSensorArmed
SoundAlarm()
endif

```

Examiner's Comments

Many responses achieved highly on this question. The question asks for a simple program to be written that checks the given variables and calls the given procedure when necessary.

Examiners were instructed to be generous with the first mark, crediting any use of selection or condition-controlled iteration. Responses may therefore have been rewarded for an attempt at this question even if their solution was not fully functional.

Centres should encourage candidates to attempt each question for precisely this reason; it is typical that a small number of marks are allocated to attempting a solution on many programming questions for

J277/02.

A significant number of responses were given 3 out of 4 marks as they misunderstood the role of operator precedence in their solution; this is detailed in the "misconception" box below.



Misconception

Where multiple conditions are used in selection, these have an order of precedence very much like BIDMAS does in mathematics; an **AND** operator will always take precedence over an **OR** operator. A **NOT** operator (not used in this question) would have even higher precedence.

This can cause problems in candidate responses. A common candidate response was:

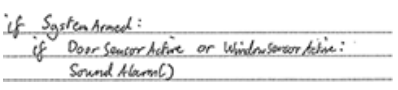
```
if SystemArmed AND
DoorSensorActive OR
WindowSensorActive then
SoundAlarm()
```


However, because the **AND** operator takes precedence, the first check done here is if the system is armed and the door sensor is active. The result of this is then evaluated with an **OR** operator to check if the window sensor is active.

This results in the alarm sounding if the window sensor is active, even if the system is not armed. This was clearly not the candidate's intention.

To fix this, candidates could have either:

- put brackets/parentheses around the `Door OR`

					<p>Window section of their response</p> <ul style="list-style-type: none"> written the response as separate checks. This could have been done in multiple ways, including nested <code>if</code> statements or repeated checks. <p>Exemplar 1</p>  <p>Exemplar 1 shows one way that full marks are achieved on this question. The candidate has used nested <code>if</code> statements to check if the system is armed, and if true, then checking if either sensor has been activated. The <code>SoundAlarm()</code> procedure is only called if both <code>if</code> statements evaluate to True.</p>
	c	i	1 mark for	1 (AO 3 1)	<p><u>Examiner's Comments</u></p> <p>Concatenation is the process of joining strings together. In OCR Exam Reference Language, this is done using the <code>+</code> symbol. Line 04 joins together <code>sensorType</code> and <code>sensorNumber</code>, assigning the concatenated result to the variable <code>sensorID</code>.</p>
		ii	1 mark from	1 (AO 3 1)	<p>Do not penalise case, spacing or minor misspellings.</p> <p><u>Examiner's Comments</u></p> <p>The vast majority of candidates are confident with identifying variable identifiers, such as <code>sensorNumber</code>. Small errors in spelling or spacing were not penalised. Candidates should be encouraged to be accurate with their namings,</p>

					particularly if these are already given to them in the question.
		iii	<p>1 mark for</p> <ul style="list-style-type: none"> Boolean 	<p>1 (AO 3 1)</p>	<p>Ignore minor misspelling.</p> <p>Accept Bool.</p> <p><u>Examiner's Comments</u></p> <p>Many candidates found this question challenging. The question relies on understanding of how selection statements operate. The answer of Boolean can be inferred from how the function return value is used.</p> <p>This use of Boolean values in selection statements also caused confusion in the previous J276 specification. It would be beneficial for centres to cover this specifically. More detail is given in the misconception box below.</p> <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <p style="text-align: center;">  Misconception </p> <p>Where Boolean values are used in selection, there is no need to compare this to a <code>True</code> or <code>False</code> value. Line 02 uses an IF statement based on the return value from a function <code>CheckSensorCode()</code>. Because the function returns a Boolean value, the IF statement is valid. Looking at another example, if the variable <code>x</code> is Boolean, then both of the following would be valid:</p> <ul style="list-style-type: none"> <code>if x == True then...</code> <code>if x then...</code> <p>In both cases, if the value of <code>x</code> is <code>True</code>, the code underneath would be executed. The second version (without comparing a Boolean value to <code>True</code> or <code>False</code>) is more elegant. However, both would be accepted as responses in an examination.</p> </div>

		iv	<p>1 mark from</p> <ul style="list-style-type: none"> • Line 01 • Line 02 • Line 03 • Line 05 	<p>1 (AO 3 1)</p> <p><u>Examiner's Comments</u></p> <p>The program code given contains two explicit calls to functions. There is one function call on line 02 for <code>CheckSensorCode()</code> and one function call on line 05 for <code>ResetSensor()</code>. However, many programming languages (including Python) implement input as a function and so examiners were instructed to also give credit where candidates identified lines 01 or 03.</p>
		v	<p>1 mark each</p> <ul style="list-style-type: none"> • Selection • Sequence 	<p>2 (AO 3 1)</p> <p>Ignore minor spelling errors / differences</p> <p>Do not accept examples (e.g. IF)</p> <p><u>Examiner's Comments</u></p> <p>The three programming constructs given in the specification are selection, sequence and iteration.</p> <p>Sequence and selection are used within this program.</p> <p>There is no use of iteration in the given program.</p> <p>Surprisingly for an AO1 question, this proved relatively challenging for candidates with many identifying other features of the code, such as inputs, function calls or variables.</p>
	d		<p>1 mark each</p> <ul style="list-style-type: none"> • <code>SELECT SensorID / SELECT *</code> • <code>FROM events</code> • <code>WHERE Length > 20 AND sensorType = "Door" /</code> <code>WHERE sensorType = "Door" AND Length > 20</code> 	<p>3 (AO 3 2c)</p> <p>Max 2 if out of order or anything extra that affects the output.</p> <p>BP1 can select multiple fields as long as <code>SensorID</code> is included.</p> <p>Ignore case. Only penalise spaces if obvious.</p> <p>Field names must be correct.</p> <p>"door" must be in quotation marks for</p>

				<p>BP3. Allow quotation marks for field names and table name</p> <p>BP3 can use == or = for equivalence.</p> <p>Allow alternative WHERE clauses that are logically correct (e.g. <code>WHERE length >=21</code>)</p> <p><u>Examiner's Comments</u></p> <p>Structured Query Language is obviously well understood by many candidates. Many high-quality responses were produced.</p> <p>Most responses were able to use SELECT and FROM appropriately to produce a logically correct response. However, the vast majority of responses missed off the requirement that only door sensors were required to be included, gaining 2 out of 3 marks in the process.</p> <p>Although a suggested response is shown in the mark scheme, any logically correct SQL that produces the required output would be accepted by examiners.</p> <p>Where a mistake was made consistently (such as using colons after the SQL keyword), this was penalised once and then FT (follow through) allowed for subsequent marks.</p>
	e	<p>1 mark each</p> <ul style="list-style-type: none"> • Define procedure SaveLogs... • ...with two valid parameters • Open file (for write/append) ... • ... using the file name passed in as parameter • Write data to file... • ...using the data passed in as parameter • Close file 	<p>6 (AO 3 2b)</p>	<p>Must be clear that answer is a procedure definition, do not credit calling procedure for BP1. Allow function definition.</p> <p>If parameters are later overwritten, do not credit BP2 but FT for BP4 and 6.</p> <p>Closing text file does not need reference to file name/object – e.g. “close file” is enough. However,</p>

				<p>if given reference must be correct.</p> <p>If code given outside of procedure, do not give BP4 and BP6</p> <p>Allow FT for multiple occurrences of same mistake (e.g. not using filename correctly for open and close)</p> <p><u>Example answer</u></p> <pre> procedure SaveLogs(data, filename) logFile = open(filename) logFile.writeLine(data) logFile.close() endprocedure </pre> <p><u>Examiner's Comments</u></p> <p>This question proved to be challenging for many candidates. The question combined defining a procedure with the use of text files.</p> <p>The tasks required were partially decomposed in the bullet points. A candidate attempting these in order would have achieved a significant number of marks.</p> <p>Candidates could also have achieved numerous marks for a partial solution (e.g. defining a procedure that didn't use text files or writing to a text file outside of a procedure) and the mark scheme was deliberately constructed to credit these responses.</p> <p>Full marks were often given where candidates appear to have had practical experience of these two techniques.</p> <p>Exemplar 2</p>
--	--	--	--	---

				<pre> procedure SaveLogs(data, fileName) file = open(fileName) file.writeLine(data) file.close() end procedure </pre> <p>Exemplar 2 shows a response that scored full marks. The procedure has been defined with multiple parameters which are then used to open the file and to write the data. The candidate has also achieved the bullet point 7 on the mark scheme (closing the file) but this wasn't necessary in this case.</p>
f	i	<p>1 mark for:</p> <ul style="list-style-type: none"> Casting / cast 	<p>1 (AO 3 2a)</p> <p><u>Examiner's Comments</u></p> <p>The use of the term "casting" to convert one data type to another is now well known and understood by candidates. This is given and referred to in the J277 specification and is essential knowledge.</p>	
	ii	<p>1 mark each to max 6</p> <ul style="list-style-type: none"> Input date and store in variable / use directly Access all seven (indexes 0 to 6) events in array / loop for each event in array Attempt at selection... ...to compare <code>date</code> input against <code>date</code> <u>in array</u> (element 0) ...adding <code>length</code> (element 3) <u>from array</u> to the total <u>if dates match</u>. Outputting <u>calculated</u> total and date in appropriate message(s) <u>at the end</u> 	<p>6 (AO 3 2b)</p> <p>BP2 can be achieved either by iteration accessing each event or manually repeating code to access each event. Must be 0 to 6, not 1 to 7.</p> <p>Allow reference to <code>events</code> (table given) or <code>arrayEvents</code> (2D array) in answer as long as used consistently.</p> <p>BP2 loop allow off by one errors (Python), looping to array length or array length – 1. Allow for each item in array or any other suitable loop.</p> <p>BP4 and BP5 allow array reference as either column major or row major.</p> <p>Output can either be once at the end or on every iteration, as long as it is output at the end.</p>	

Only give output mark if attempt made to **calculate** total within the algorithm.

Do not penalise capitalisation or minor misspellings of variable names.

Example answer 1

```
total = 0
date = input("Please enter date")
for count = 0 to events.length-1
if events[0, count] == date then
total = total + events[3,count]
endif
next count
print("There were " + total + " events on " + date)
```

Example answer 2

```
total = 0
date = input("Please enter date")
for item in events:
if item[0] == date then
total = total + item[3]
endif
next count
print("There were " + total + " events on " + date)
```

Examiner's Comments

The final question in Section B is expected to be a high demand question.

The techniques required (iteration through a 2D array, selection, keeping a running total of times) are within the specification but it is acknowledged that the level of challenge was high.

Examiners were instructed to give marks for an attempt at a solution (as with previous questions).

For this question marks were given for:

- any attempt at selection
- any solution that accessed each element in the given array, even if this was via a manual process.

Therefore, many candidates gained multiple marks for an attempt that only partially solved the problem.

A significant number of candidates were able to create a solution that fully met the requirements of the question. This was often done in an elegant and efficient manner.

This is extremely pleasing and shows excellent understanding and significant experience of practical programming.


Exemplar 3

```
total = 0
date = input("Enter a date")
total = 0
count = 0
for count = 0 to arrayEvents.length
    if arrayEvents[count] == date then
        total = total + arrayEvents[count]
    endif
endfor
print("Sensors were activated for", total, "seconds on", date)
```

Exemplar 3 shows a high scoring response. A date has been asked for as input which has then been used to compare to each element at position 0 in the array.

Where any of these match, the total variable is updated to keep a running

					<p>total of the corresponding element at position 3 in the array.</p> <p>After each element has been checked, the total and date are output in a suitable message.</p> <p>This is not the only method by which a response could be given full marks but is perhaps the most common.</p>															
			Total	30																
1 2	a		<p>1 mark per correct row</p> <table><thead><tr><th>OCR Reference Language code</th><th>Selection</th><th>Iteration</th></tr></thead><tbody><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 = 0 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</pre></td><td>✓</td><td></td></tr></tbody></table>	OCR Reference Language code	Selection	Iteration	<pre>for i = 1 to 10 print(i) next i</pre>		✓	<pre>whilescore != 0 playgame() endwhile</pre>		✓	<pre>if playerHit() then score = 0 endif</pre>	✓		<pre>switch bonus: case 0: score = 9 case 1: score = 7 case 2: score = 5</pre>	✓		4 (AO 2 1b)	<p>No mark given if both boxes in a row ticked.</p> <p>Accept any response (ticks, crosses, etc) that clearly indicates candidate's choice.</p> <p><u>Examiner's Comments</u></p> <p>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.</p>
OCR Reference Language code	Selection	Iteration																		
<pre>for i = 1 to 10 print(i) next i</pre>		✓																		
<pre>whilescore != 0 playgame() endwhile</pre>		✓																		
<pre>if playerHit() then score = 0 endif</pre>	✓																			
<pre>switch bonus: case 0: score = 9 case 1: score = 7 case 2: score = 5</pre>	✓																			

			endswitch		
		b	<ul style="list-style-type: none"> score = score + 1 / score +=1 / score++ 	<p>1 (AO 3 2b)</p>	<p>Allow other logically correct answers that result in score increasing by one and being overwritten. Do not accept <code>score + 1 / score = +1</code></p> <p>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"</p> <p>Ignore any superfluous code that does not affect the outcome</p> <p><u>Examiner's Comments</u></p> <p>Most candidates were able to temporarily add one to the value of score (by using code such as <code>score + 1</code>). Fewer were able to assign this new value to the variable of score and therefore fully answer the question. Answers such as <code>score = score + 1</code>, <code>score++</code> or <code>score += 1</code> 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.</p> <p> Assessment for learning</p> <p>One notable response that was not allowed was <code>score =+1</code>. This statement assigns the value of 'positive 1' to score, overwriting the previous value held.</p>

					This is a good example of the precision required to gain marks. Candidates with practical programming experience are more likely to recognise this.															
			Total	5																
1 3		i	<ul style="list-style-type: none">Convert/change one data type to anotherLine 03 / 3 / three	2 (AO 1 1b) (AO 2 2b)	<p>Do not accept "change to string" - this is the use in this example but not a definition.</p> <p><u>Examiner's Comments</u></p> <p>Many candidates correctly defined casting as changing data from one data type to another. Some candidates defined this term as changing a variable from an integer to a string, which is only one example of what can be done and not a definition.</p> <p>The majority of candidates then gave the correct line number (line 03) for there this was shown the example code given.</p>															
		ii	<ul style="list-style-type: none">Kofi2021 as staffID on line 03Kofi2021x as staffID on line 05Kofi2021xx as staffID on line 05ID Kofi2021xx output on line 07 as first and only output	4 (AO 3 2c)	<p>Max 2 if incorrect order. Ignore misspelling of Kofi</p> <p>Penalise lack of / errors with line numbers once then FT. Ignore capitalisation. Ignore additional lines unless outcome impacted.</p> <p>staffID does not have space in. Output does have a space in. Penalise spaces once then FT. Do not penalise unless obvious.</p> <p>Quotes around answer is OK, but do not allow quotes around partial answers, e.g. "ID" Kofi2021xx is incorrect.</p> <table><tr><th>Line number</th><th>surname</th><th>year</th><th>staffID</th><th>Output</th></tr><tr><td>01</td><td>Kofi</td><td></td><td></td><td></td></tr><tr><td>02</td><td></td><td>2021</td><td></td><td></td></tr></table>	Line number	surname	year	staffID	Output	01	Kofi				02		2021		
Line number	surname	year	staffID	Output																
01	Kofi																			
02		2021																		

					<table><tr><td>03</td><td></td><td></td><td>Kofi2021</td><td></td></tr><tr><td>05</td><td></td><td></td><td>Kofi2021x</td><td></td></tr><tr><td>05</td><td></td><td></td><td>Kofi2021xx</td><td></td></tr><tr><td>07</td><td></td><td></td><td></td><td>ID Kofi2021xx</td></tr></table>	03			Kofi2021		05			Kofi2021x		05			Kofi2021xx		07				ID Kofi2021xx
03			Kofi2021																						
05			Kofi2021x																						
05			Kofi2021xx																						
07				ID Kofi2021xx																					
					<p><u>Examiner's Comments</u></p> <p>This question asked candidates to trace through a given algorithm to show the value of three variables at various points in the algorithm.</p> <p>The algorithm itself was relatively simple. It used condition-controlled iteration to repeat while the length of the username was less than 10 characters.</p> <p>Most candidates gained the first 2 marks for the initial changes to <code>staffID</code>. However few candidates were able to trace through the iteration and conclude that the final username should end up as ID Kofi2021xx.</p> <p>Marking this question considered the spaces within the username at various points. The algorithm results in one space only, in between ID and Kofi2021xx. Where extra spaces appeared or were missed, this was penalised. However, examiners were instructed to give clear benefit of doubt, and to only do this if the space was clearly present/missing.</p> <p>It is important to understand that "ab" and "a b" are two strings that are not the same. This level of precision should be encouraged within GCSE Computer Science. Experience of practical programming will help reinforce the impact of spaces within programming and algorithms.</p>																				
			Total	6																					

1 4		i	<ul style="list-style-type: none"> • Multiplication • Division 	2 (AO 1 1a)	<p>Accept other correct answers that mean the same Accept floor / integer division / division with no remainder (Python v2.x)</p> <p><u>Examiner's Comments</u></p> <p>This question was answered well by the majority of candidates.</p>
		ii	<ul style="list-style-type: none"> • high-level • stops / crashes • no • executable • without 	5 (AO 1 1b) (AO 2 1b)	<p>Ignore spelling errors.</p> <p><u>Examiner's Comments</u></p> <p>This question was answered well by the majority of candidates.</p>
			Total	7	
1 5	a	i	<ul style="list-style-type: none"> • Integer • String 	2 (AO 3 2a)	<p>Accept other valid data types from high-level languages (e.g. byte, short for integers)</p> <p>Do not accept descriptions (e.g. "whole number", "text"). Do not accept "character(s)" for string.</p>
		ii	<ul style="list-style-type: none"> • stayComplete 	1 (AO 3 2a)	<p>Ignore spaces or misspelling as long as recognisable.</p> <p><u>Examiner's Comments</u></p> <p>These questions tested candidates' knowledge of data types and it was clear that this knowledge was well understood. The majority of candidates were able to correctly identify suitable data types in section (i) and identify <code>stayComplete</code> as the field that would be stored as a Boolean data type.</p>
		iii	<ul style="list-style-type: none"> • <code>SELECT FirstName, Surname, Nights, Room, StayComplete / SELECT*</code> • <code>FROM TblBookings</code> • <code>WHERE</code> • <code>Nights > 1 / Nights >= 2 / Nights BETWEEN 2 AND 5.</code> 	4 (AO 3 1) (AO 3 2c)	<p>Order of fields for BP1 not important but must show all fields and be separated by commas.</p> <p>Ignore capitalisation and spacing. Spelling must be correct. Ignore quotes around numeric values or field/table names.</p>

				<p>Allow other logically valid SQL statements. Check with TL if required.</p> <p>Ignore reference to <code>stayComplete</code> or other valid SQL code that would not affect output.</p> <p>Max 3 if in wrong order or if includes any extra invalid code</p> <p><u>Examiner's Comments</u></p> <p>This question tested candidates' ability to refine and rewrite incorrect code given to them. It is important to note that although the SQL statement as a whole is incorrect, not all components are incorrect; in this case, the <code>FROM</code> clause is correct and candidates who made no change to this line were credited.</p> <p><code>SELECT ALL</code> is invalid SQL and should have been written to instead include all fields from the table, separated by commas <code>SELECT *</code> was equally accepted as a suitable response.</p> <p><code>IF</code> is not a valid SQL keyword and needs to be replaced with <code>WHERE</code>. The criteria for this statement was also incorrect. The comparison symbol is incorrect and should read <code>Nights > 1</code>.</p> <p>Most candidates gained some marks on this question. The most common response corrected the criteria and not modifying the <code>FROM</code> clause.</p>
	b i	<ul style="list-style-type: none"> Checks that both <code>firstname</code> and <code>surname</code> are not empty... Checks that <code>room</code> is either "basic" or "premium"... Checks that <code>nights</code> is between 1 and 5 (inclusive)... ...Outputs "NOT ALLOWED" (or equivalent) if <u>any</u> of the 3 checks are invalid (must check all three) 	5 (AO 3 2a)	<p>Must have some attempt at <u>all three checks</u> to give output mark(s). Check for <code>nights</code> must check both upper and lower limits.</p> <p>Iteration can be used as validation if input repeatedly asked for until valid answer given.</p>

		<ul style="list-style-type: none"> ...Outputs "ALLOWED" (or equivalent) <u>only</u> if all three checks are valid (must check all three) <p><i>Note : output marks are given for if entire system produces the correct output. For example, If a user enters a valid name and room but an invalid number of nights, the system should say "NOT ALLOWED" (or equivalent). If this works and produces the correct response no matter which input is invalid, BP4 should be given.</i></p> <p><i>The same process holds for the valid output - if (and only if) three valid inputs results in an output saying "ALLOWED" (or equivalent), BP5 should be given. Do not give this if ALLOWED is printed when (for example) two inputs are valid and one is invalid.</i></p> <p><i>For any output marks to be given, a sensible attempt must have been made at all three checks. These may not be completely correct (and may have been penalised in BPs 1 to 3) but should be enough to allow the FT marks for output.</i></p>	<p><u>Do not accept</u> logically incorrect Boolean conditions such as <code>if firstname or surname == ""</code></p> <p>Do not accept \geq or \leq for $>=$, $<=$. Ignore capitalisation</p> <p>e.g.</p> <pre>valid = True if firstname == "" <u>or</u> surname == "" then valid = False end if if room != "basic" <u>and</u> room != "premium" then valid = False endif if nights < 1 <u>or</u> nights > 5 then valid = False endif if valid then print("ALLOWED") else print("NOT ALLOWED") endif</pre> <p>BP1 to 3 can check for valid or invalid inputs. . Pay particular attention to use of AND / OR. Only give marks for output if these work together correctly.</p> <p>Example above shows checking for invalid data. Checks for valid data equally acceptable Examples shown below:</p> <ul style="list-style-type: none"> <code>if firstname != "" <u>and</u> surname != ""</code> <code>if room == "basic" <u>or</u> room == "premium"</code> <code>if nights >= 1 <u>and</u> nights <= 5</code> <p><u>Examiner's Comments</u></p> <p>This question stretched the understanding of even highly-</p>
--	--	--	---

achieving candidates and it was not uncommon to see low scoring responses.

Misunderstanding of Boolean operators (AND and OR) within selection (IF) statements was something that affected a lot of candidate responses.

As this question was in Section B, candidates needed to respond in OCR Exam Reference Language or a high-level language. Responses must be logically correct to gain the marks. As each check is two individual checks that both need to pass, responses can quickly get relatively complicated.

As can be seen from the mark scheme, advice and examples were given to examiners to make sure that candidates who were able to successfully navigate this logic chain were credited.



Misconception

Checking whether a room is either basic or premium can be done in multiple ways. Candidates can either check for the positive (i.e. check that it is either basic or premium) or check that for the negative (i.e. check whether it is something else). However, there are many common errors that were seen :

- `IF room == "basic" or "premium"` is **incorrect** as the second part of the statement is not evaluated against anything. This was perhaps the most common mistake.
- `IF room == "basic" or room == "premium"` is

				<p>correct and checks for validity.</p> <ul style="list-style-type: none"> • IF room == basic or room == premium is incorrect as the lack of string delimiters means that basic and room would be treated as variables rather than strings. • IF room != "basic" or room != "premium" is also incorrect. This checks for invalid input but because or is used, only one condition needs to be True for the whole statement to be True. This means that if basic is entered, it would be classed as invalid (as it isn't premium) and vice-versa. There is no way for any entry in this example to be classes as valid. • IF room != "basic" and room != "premium" is correct. This checks for invalid inputs but needs both conditions to be True. <p>The same explanation follows for the other two necessary checks.</p> <p>Exemplar 3</p> <pre> if first Name == "" OR surname == "" then print("NOT ALLOWED") else if room != "basic" AND room != "premium" then print("NOT ALLOWED") else if night < 1 OR night > 5 then print("NOT ALLOWED") else print("ALLOWED") end if end if end if end if </pre> <p>This exemplar shows a fully correct response. The candidate checks for invalid responses and correctly uses Boolean operators to check multiple criteria at each step. If any check returns True, "Not allowed" is printed</p>
--	--	--	--	--

					<p>and the program ends. Efficient use of <code>if ... else ...</code> means that the next check only proceeds if the previous check returns False.</p> <p>If all three checks return False, the final else is triggered to print “Allowed”.</p> <p>It must be noted that this is only one way of achieving full marks. An equivalent program that checks for valid responses at each turn would also be possible. Candidates should be encouraged to use whatever structure they feel is sensible. If a response can logically be followed then it will achieve high marks.</p>												
		ii	<ul style="list-style-type: none">• Normal• 1 or 5 (<i>not 0 or 6 as says allowed</i>)• Any numeric value except 1 to 5 / any non-numeric input (e.g. "bananas")	3 (AO 3 2c)	<p>Allow other descriptions that mean normal (e.g. valid / typical / acceptable)</p> <table><tr><th>Test data (number of nights)</th><th>Type of test</th><th>Expected output</th></tr><tr><td>2</td><td>Normal</td><td>ALLOWED</td></tr><tr><td>1 / 5</td><td>Boundary</td><td>ALLOWED</td></tr><tr><td>e.g. 7</td><td>Erroneous/Invalid</td><td>NOT ALLOWED</td></tr></table> <p><u>Examiner’s Comments</u></p> <p>This question was answered well by the majority of candidates.</p>	Test data (number of nights)	Type of test	Expected output	2	Normal	ALLOWED	1 / 5	Boundary	ALLOWED	e.g. 7	Erroneous/Invalid	NOT ALLOWED
Test data (number of nights)	Type of test	Expected output															
2	Normal	ALLOWED															
1 / 5	Boundary	ALLOWED															
e.g. 7	Erroneous/Invalid	NOT ALLOWED															
		c i	<ul style="list-style-type: none">• Function header for newPrice...• ...taking (at least) two parameters• ...correctly calculates price based on parameters (if present) <u>within function</u> .../• ... returns this calculated price	4 (AO 3 2b)	<p>BP1 must be clear that a new function is being defined. E.g. <code>function</code> / <code>def</code> keyword. Allow FT for subsequent marks if not present.</p> <p>Ignore any code outside attempt at function definition.</p> <p>Ignore additional parameters. Ignore inputs or additional code as long as these do not overwrite parameters or affect operation of function.</p>												

If inputs used instead of parameters, FT for BP3. Allow use of `else` for second room type in BP3.

Attempt at calculation needed to award BP4. Must return (not output) value. Return can be done e.g. in VB by assigning to function name (e.g. `newPrice = price`)

e.g.

```
function newPrice(nights,
room)

    if room == "basic" then

        if room== 60 * nights
then

            elseif room ==
"premium" then

                price = 80 * nights

            endif

        return price

    endifunction
```

Examiner's Comments

Defining functions appeared to be a concept that candidates did not fully understand.

Where a candidate did not attempt to define a function and instead simply calculated the price needed, very few (if any) marks were available.

Successful responses could have been constructed from any suitable function definition keyword such as `function` (OCR ERL, VB, JavaScript, etc), `def` (Python) or others. Answers in C#, Java or other languages referring to methods were also accepted.

		ii	<ul style="list-style-type: none"> • Call function newPrice... • ...with <u>("premium", 5)</u> as parameters • ... Output returned value 	<p>Order of parameters not important</p> <p>"premium" must use string delimiters (e.g. "quotes")</p> <p>e.g.</p> <pre>print(newPrice("premium", 5))</pre> <pre>x = newPrice(5, "premium")</pre> <pre>print(x)</pre> <p>Do not allow function definitions for BP1</p> <p>Ignore capitalisation of newPrice</p> <p>Candidate could store returned value in a variable and then print this, or store parameters in variables before passing in - these are all acceptable</p> <p>Ignore any superfluous code given</p> <p>Do not credit answers where newPrice is overwritten prior to use.</p> <p>Ignore spaces. Allow function call if brackets missing (e.g. newprice instead of newprice())</p> <p><u>Examiner's Comments</u></p> <p>Even if candidates were not able to create a function, this question was independent to (i) and so marks were available for simply using the function to output a value.</p> <p>Candidate found this question challenging. Many candidates called the function but most did not understand that the room type was a string and so required string delimiters (e.g. quotation marks) around the parameter.</p> <p>Where candidates defined local</p>	<p>3 (AO 3 2b)</p>
--	--	----	--	--	--------------------------------

					variables, assigned the values needed to the variables and then passed these into the function call were accepted.
	d		<ul style="list-style-type: none"> Inputs hours AND electric (two separate inputs), storing or using these. Checks if car is electric (IF/Select statement)... ... correctly calculates and outputs price (hours * 2 / price / 2) for electric ... correctly calculates and outputs price (hours * 4 / electric price * 2) for non-electric Attempt at repetition of BP1 to 4... ...until 0 hours entered 	<p>6 (AO 3 2c)</p>	<p>Initialisation of price and hours not necessary, but if present hours must be non-zero for BP6 to be given.</p> <p>BP5 must include all points attempted. Can still be credited if any of BP1 to 4 not attempted / incorrect.</p> <p>BP6 can be given as FT even if BP5 (loop) is in the wrong place / does not include all required code</p> <p>BP6 could be achieved as repeated function calls / recursion</p> <p>Initial input outside of loop that is then <u>also</u> included within loop is fine. For example, input of hours outside of loop but input is then repeated again at end of loop.</p> <p>Do not accept <code>while hours > 0</code> (could be -1)</p> <p>Do not penalise answers where 0 is output when loop exits</p> <p>e.g.</p> <pre>while hours != 0 hours = input("Enter hours") electric = input("enter Y for electric or N") if electric == "Y" then price = hours * 2 elseif electric == "N" then price = hours * 4 endif print(price) endwhile</pre> <p><u>Examiner's Comments</u></p>

					<p>This question was relatively well answered by candidates.</p> <p>Candidates were generally able to create suitable high-level program code to calculate and output the total price based on the information given.</p> <p>Many candidates ignored the requirement to repeat until 0 was entered; in this case, 4 out of the 6 marks were still available.</p> <p>To achieve marks for iteration it needed to both repeat the correct parts of the program and correctly terminate as per the requirements given.</p> <p>A typical mistake was to repeatedly calculate the price but not ask afresh for new inputs.</p>															
			Total	28																
1 6			<table><tr><td>A</td><td>B</td><td>P</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td>1</td></tr><tr><td></td><td></td><td>1</td></tr><tr><td></td><td></td><td></td></tr></table>	A	B	P						1			1				2 (AO 1 1b)	<p>1 mark for each correct answer in table</p> <p>‘True’ or ‘T’ are also credit worthy.</p>
A	B	P																		
		1																		
		1																		
			Total	2																
1 7			<ul style="list-style-type: none">• SELECT StudentName, Subject, Grade• FROM Results• WHERE Subject = "Art"	1 (AO 1 1b) 2 (AO 3 2a)	<p>Correct Answer Only</p> <p>Accept SELECT *</p>															
			Total	3																
1 8			<ul style="list-style-type: none">• freeseats called with "Red"	2	<code>redseats = freeseats("Red")</code>															

			<ul style="list-style-type: none"> ...<u>returned value</u> assigned to variable <u>redseats</u> 		"Red" must use suitable string delimiters (e.g. speech marks) if directly passing the string. Do not penalise case.
			Total	2	
1 9	a		<p>One mark per correct choice</p> <ul style="list-style-type: none"> SELECT ItemCode, ItemName FROM tblStock WHERE Price >=60 	3	Accept other markings that indicate a choice has been made (e.g. a cross, etc)
	b	i	<p>One mark per bullet point, in the correct place</p> <ul style="list-style-type: none"> size / len(discountcodes-1) code price / newprice [x,1] / [x][1] return newprice / checkdiscount = newprice 	5	<p>e.g.</p> <pre>function checkdiscount(price, code) newprice = price size = len(discount)-1 for x = 0 to size if discount[x,0] == code then newprice = price - discount[x,1] endif next return newprice endfunction</pre>
		ii	<p>One mark per bullet point, maximum 2 marks</p> <ul style="list-style-type: none"> newprice size x 	2	<p>Do not penalise capitalisation</p> <p>Accept price, code, discount</p>
		iii	<ul style="list-style-type: none"> asks for price and discount code to be input ...passes both to the checkdiscount() function as parameters... ...stores / uses returned value calculates total of all prices entered/returned repeats until 0 is entered as <u>price</u> outputs <u>calculated total</u> 	6	<p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural language.</p> <p>BP3 allow total of prices entered as FT if candidate does not achieve BP2</p> <p>e.g.</p> <pre>total = 0</pre>

					<pre>do price = input("Enter a price") code = input("Enter a discount code") newprice = checkdiscount(price, code) total = total + newprice until price == 0 print(total)</pre> <p>alternative example</p> <pre>total = 0 price = 1 while price != 0 price = input("Enter a price") code = input("Enter a discount code") total = total + checkdiscount(price, code) endwhile print(total)</pre>
			Total	16	
2 0	a		<p>1 mark for naming the example and 1 mark for an example related to that method</p> <p>E.g</p> <ul style="list-style-type: none"> • Comments/annotation... • ...E.g. any relevant example, such as line 4 checks the input is valid • Indentation... • ...E.g. indenting within IF statement • Using constants... • ...E.g. π 	4 (AO 2 1b)	
	b		<ul style="list-style-type: none"> • radius • area 	2 (AO 1 1b)	1 mark per bullet up to a maximum of 2 marks.
	c	i	<ul style="list-style-type: none"> • 3.142 • 2 • 1 	1 (AO)	1 mark for one correct identification.

			<ul style="list-style-type: none"> 30 	2 1a)	
		ii	<ul style="list-style-type: none"> The number does not need to be changed while the program is running The number can be updated once and it updates throughout 	1 (AO 1 1a)	Maximum of 1 mark.
		d	<ul style="list-style-type: none"> HAS been used HAS been used HAS NOT been used 	3 (AO 2 1b)	
			Total	11	
2 1		a	<p>Integer (1)...</p> <ul style="list-style-type: none"> ...number of seconds not important (1) ... level of accuracy not needed so round to nearest minute (1) ...using a decimal to store seconds (0-60) is not appropriate (1) <p>Real (1)...</p> <ul style="list-style-type: none"> ... number of seconds may be important (1) ... allows parts/fractions to be stored over integers (1) 	1 (AO 3 2a) 1 (AO 3 1)	<p>One mark for appropriate data type identified.</p> <p>One mark for appropriate justification linked to the data type chosen.</p>
		b	<pre>print (minsPlayed[0,4])</pre>	1 (AO 3 2b)	<p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural English.</p> <p><code>print</code> may be a suitable output command word that could be found in a HLL e.g. <code>print</code> (Python), <code>console.writeline</code> (VB), <code>cout</code> (C++)</p> <p>The array elements may be</p>

					accessed together [0, 4] (VB.NET) or separately [0][4] (Python)																													
	C		<table><tr><td></td><td>x</td><td>y</td><td>output</td></tr><tr><td>MP1</td><td>15</td><td>0</td><td></td></tr><tr><td rowspan="2">MP2</td><td>14</td><td>1</td><td></td></tr><tr><td>12</td><td>2</td><td></td></tr><tr><td rowspan="3">MP3</td><td>9</td><td>3</td><td></td></tr><tr><td>5</td><td>4</td><td></td></tr><tr><td>0</td><td>5</td><td></td></tr><tr><td>MP4</td><td></td><td></td><td>5</td></tr></table>		x	y	output	MP1	15	0		MP2	14	1		12	2		MP3	9	3		5	4		0	5		MP4			5	4 (AO 3 2c)	one mark for first row one mark for row 2 and 3 one mark for rows 4, 5, and 6 one mark for the correct output (the only value in the output column, in any position)
	x	y	output																															
MP1	15	0																																
MP2	14	1																																
	12	2																																
MP3	9	3																																
	5	4																																
	0	5																																
MP4			5																															
			Total	7																														