


Mark scheme

| Question | | | Answer/Indicative content | Marks | Guidance |
|----------|--|--|--|------------|--|
| 1 | | | <p>2 marks max per group</p> <ul style="list-style-type: none"> Meaningful identifiers / meaningful variable names ...to describe/show what they store / purpose of variable An example of a meaningful variable identifier <u>for this algorithm</u> 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 <u>for this algorithm</u> Use of subroutines ...to reuse blocks of code / make code easier to follow An example of a subroutine <u>for this algorithm</u> 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 <u>for this algorithm</u> (e.g. store 512 as a constant) | 4 (AO2) | <p>Do not accept "what variables do" – incorrect verb, variables store/hold data.</p> <p>BOD notes (and alternatives) for comments. Do not allow instructions.</p> <p>Do not allow indentation (already done in program given)</p> <p>Allow whitespace / blank lines (same expansions as comments)</p> <p>Do not award expansion without being clear which method is being discussed. "Makes it easier to understand" by itself is TV.</p> <p><u>Examiner's Comments</u></p> <p>It was pleasing that the majority of candidates understood the term maintainability and were able to suggest suitable improvements on a generic level, such as improving the naming of variables and adding comments. Better responses that achieved full marks were able to apply this to the code given, such as suggesting more suitable variable names.</p> <p>Candidates who suggested adding indentation were not credited with marks as this has clearly already been done in the code given and thus would not be an improvement.</p> <p>Key point – generic versus context driven responses</p> <p>Where a question gives a context or scenario (such as data or a</p> |
| | | | | | |

| | | | | |
|---|---|----|--|---|
| | | | | <p>program being given), it is important that candidates refer to this context in their response if they are hoping to achieve full marks. For this question, the question text was:</p> <p>‘Describe two ways to improve the maintainability of this algorithm’</p> <p>This is a very different question from ‘Describe two ways to improve the maintainability of an algorithm’ where a generic response would suffice.</p> |
| | | | Total | 4 |
| 2 | a | i | <ul style="list-style-type: none"> String Integer Real / Float | <p>3 (AO3)</p> <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 (AO3)</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</p> |

| | | | | |
|--|--|--|--|--|
| | | | | <p>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 challenging mark was certainly the use of <code>count</code> as the index of the <code>theTeam</code> array, with very few candidates correctly identifying this as the missing element.</p> <p> Assessment for learning</p> <p>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.</p> <p>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.</p> |
|--|--|--|--|--|

| | b | <ul style="list-style-type: none"> • javelinThrow set to 14.3 on line 01 <u>and</u> • 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) <p><u>Example</u></p> <table border="1" data-bbox="411 1039 801 1158"> <thead> <tr> <th>Line number</th><th>javelinThrow</th><th>yearGroup</th><th>score</th><th>Output</th></tr> </thead> <tbody> <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> </tbody> </table> <p>Answer may include lines where no changes or output happens (i.e. lines 3, 4, 5, 7, 8, 9, 10, 12).</p> <p>Where variable doesn't change, current value may be repeated on subsequent lines.</p> | Line number | javelinThrow | yearGroup | score | Output | 01 | 14.3 | | | | 02 | | 10 | | | 06 | | | 2 | | 11 | | | 4 | | 13 | | | | The score is 4 | 4 (AO3) | <p>Max 3 if in wrong order or additional (incorrect) changes. Penalise line numbers once then FT.</p> <p>Allow FT for BP4 for current value of <code>score</code>.</p> <p>BP4 must <u>not</u> include comma. Ignore superfluous spaces. Ignore quotation marks.</p> <p>Treat any entry in output column as an output, even if "x", "-" or "0".</p> <p><u>Examiner's Comments</u></p> <p>Trace tables have appeared multiple times in J277 examination papers now and candidates are hopefully familiar with the expectations, which are consistent across series.</p> <p>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.</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> |
|-------------|--------------|---|-------------|---|-----------|-------|--------|----|------|--|--|--|----|--|----|--|--|----|--|--|---|--|----|--|--|---|--|----|--|--|--|----------------|------------|--|
| Line number | javelinThrow | yearGroup | score | Output | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | 14.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 02 | | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 06 | | | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | The score is 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | c i | <ul style="list-style-type: none"> • inputs a value from the user <u>and</u> <u>stores/uses</u> • checks min value (≥ 40.0 / < 40) | 4 (AO3) | <p>Answers using AND/OR for BP2 and BP3 must be logically correct e.g. if height ≥ 40 and</p> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|--|--|--|--|
| | | <ul style="list-style-type: none"> • checks max value (≤ 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><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> |
|--|--|--|--|

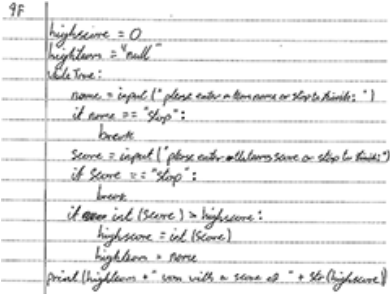
| | | | | |
|--|---|----|--|---|
| | | | | One final common problem involved the use of greater than or equal to signs (and also less than or equal to). The common signs used in mathematics are not available on a typical keyboard and so would not be allowed in Section B of this paper. Instead, >= or <= should be used, and these should be the correct way around. |
| | | 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 (AO3)</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"> TeamName only in first space TblResult in second space WHERE ...YearGroup = 11 | <p>4 (AO3)</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 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</p> |

| | | | | |
|---|---|---|------------|---|
| | | | | 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. |
| 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 (AO3) | <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 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</p> |


| | | | | |
|--|----|---|--------------------|--|
| | | | | <p>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 (AO3)</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 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</p> |


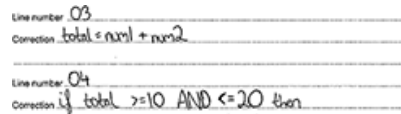
| | | | | |
|---|--|---|--------------------|--|
| | | | | <p>sensible (such as showing student IDs instead of names). Where the context of the sports day was used, candidates were generally successful in achieving this mark.</p> <p>Decomposition use was more tricky to correctly identify, as many candidates 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... • ...to enter team/score until "stop" entered • Calculates highest score • Calculates winning team name... • ...Outputs highest score and team name <p><u>Example 1</u></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><u>Example 2 (alternative)</u></p> | <p>6 (AO3)</p> | <p>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?")</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</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>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>input 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 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</p> |
|--|--|---|---|

| | | | | |
|--|--|--|--|--|
| | | | | <p>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> <div><pre>9F highscore = 0 highlow = "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) highlow = name print(highlow + " won with a score of " + str(highscore))</pre></div> <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</p> |
|--|--|--|--|--|

| | | | | |
|---|---|--|---|--|
| | | | | <p><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 | a | | <p>Max 1 mark for definition that is clearly different from a logic error.</p> <ul style="list-style-type: none"> • (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 <p>Suitable example for 1 mark, e.g.</p> <ul style="list-style-type: none"> • misspelling key word (e.g. <code>printt</code> instead of <code>print</code>) • 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. <code>print = 3</code>) • Incorrect capitalisation of keywords (e.g. <code>P_rint</code> instead of <code>print</code>) • Incorrect indentation (of code blocks) • Missing concatenation (e.g. <code>print(score x))</code> | <p>2 (AO1)</p> <p>BOD code/program etc for BP1</p> <p>Do not allow answers linked to data types.</p> <p>"incorrect grammar" by itself is NE</p> <p>Do not allow "stop working", "does not work", etc – TV.</p> <p>Do not accept <u>missing</u> quotation marks e.g. <code>print(hello)</code> (could be a variable name)</p> <p>BOD given code that could cause a syntax error in a high-level language.</p> <p><u>Examiner's Comments</u></p> <p>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.</p> <p>The second part did ask for an example and generous interpretation was asked of</p> |

| | | | | |
|--|---|---|------------|--|
| | | | | <p>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.</p> <p> Misconception</p> <p>A syntax error is a mistake with the rules/grammar of the programming language that means the program will not run/execute or compile.</p> <p>Code such as <code>print(temp)</code> would not necessarily be a syntax error because <code>temp</code> could plausibly be a variable.</p> |
| | b | <p>1 mark each</p> <ul style="list-style-type: none"> line 03 <code>total = num1 + <u>num2</u></code> Line 04 <code>if total >= 10 and total <=20</code> then <p>Allow other logical equivalent code e.g. <code>total = int(num1) + int(num2) if 10 <= total <= 20</code></p> | 4 (AO3) | <p>Allow other logical corrections that will fix the problem identified and does not introduce any further errors.</p> <p>Allow descriptions of changes as long as clear <u>exactly</u> what will change. Do not allow ambiguous descriptions of changes to code.</p> <p>Ignore missing <code>then</code> from line 04.</p> <p>Ignore capitalisation.</p> <p><u>Examiner's Comments</u></p> <p>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.</p> |


| | | | | |
|---|--|---|------------|--|
| | | | | <p>The correction to line 04 was commonly done incorrectly due to the need to check multiple values.</p> <p> Misconception</p> <p>Where multiple values are required to be checked in a selection statement, a line such as :</p> <pre>if total >= 10 and <= 20 then</pre> <p>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 :</p> <pre>if total >= 10 and total <= 20 then</pre> <p>Alternatives that work in high-level languages would also obviously be accepted.</p> <p>Exemplar 1</p>  <p>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.</p> |
| | | Total | 6 | |
| 4 | | Only 1 method asked for. Could be name and description/example or description and example | 2 (AO1) | Allow validation / input sanitisation / passwords as expansion of |

| | | <ul style="list-style-type: none">• 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) | | <p>anticipating misuse.</p> <p>Allow mark for description with no / incorrect name</p> <p>Allow any 2 points from mark scheme as long as clearly linked to a single defensive design method.</p> <p><u>Examiner's Comments</u></p> <p>The specification (Section 2.3.1) lists multiple ways that defensive design could be used in a program and any of these, plus other sensible options, were allowed as an acceptable response. The describe command word then required candidates to add further detail to obtain a second mark, in this case by describing how it could be used, either generically or as a specific example. Many points on the mark scheme crossed over with each other, such as a validation example being a sensible expansion for anticipating misuse, and hence two marks were able to be obtained in multiple ways.</p> | | | | | | | | | | | | |
|---------------------|-------------------|--|---------------|---|---------------------|--------------|-----------------|---|--------|------|----|----------|------|----|-------------------|-------|
| | | Total | 2 | | | | | | | | | | | | | |
| 5 | | <p>One mark per bullet point</p> <ul style="list-style-type: none">• Any value between 0 and 20 (e.g. 4)• True• Invalid / erroneous / sensible alternative• False | 4 (AO3 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 | | | | | | | | | | | | | | |
| | | Total | 4 | | | | | | | | | | | | | |
| 6 | a | 1 mark each: Syntax error | 2 (AO1 1a) | Question asks for a definition. Examples may strengthen the response but are not acceptable by | | | | | | | | | | | | |

| | | | | |
|--|---|---|------------------|---|
| | | <ul style="list-style-type: none"> Error in the rules/grammar (of the program language). Program does not (fully) run / translate / execute / start (BOD) <p>Logic error</p> <ul style="list-style-type: none"> Produces incorrect / unexpected result/output Program runs/does not crash | | <p>themselves.</p> <p>Do not allow “error/problem in the code, does not work / does not do what designed/intended to do” for either, this applies to both.</p> <p>“Error in the syntax” or “error in the logic” are NE even with examples</p> <p><u>Examiner’s Comments</u></p> <p>This question was answered extremely well by most candidates. These candidates correctly defined the terms given.</p> <p>Candidates must understand that an example is not the same as a definition. For example, a misspelling of a command such as <code>print</code> would of course be a syntax error but this is not a definition; many other issues would cause a syntax error.</p> <p>Other incorrect responses included generic responses that could apply to either term, for example: “a mistake in the program” or “where the computer doesn’t understand the code”.</p> |
| | b | <p>Line number</p> <ul style="list-style-type: none"> 02 <p>Correction</p> <ul style="list-style-type: none"> <code>for scoreCount = <u>0</u> to scores.length - 1</code> <p>Line number</p> <ul style="list-style-type: none"> 03 <p>Correction</p> <ul style="list-style-type: none"> <code>total = scores[scoreCount] + total</code> | 4 (AO3 2c) | <p>1 mark for each line number correctly identified. 1 mark for each correction. Correction must match line number.</p> <p>If wrong line number, do not mark correction. If no line number, mark correction only.</p> <p>Do not penalise if response removes <code>-1</code> from <code>scores.length</code> as long as it starts at 0.</p> <p>Do not penalise potential off by 1 errors for looping (Python).</p> <p>Do not penalise case or minor</p> |

| | | | |
|--|--|---|---|
| | | <ul style="list-style-type: none"> • <code>total = total + scores[scoreCount]</code> • <code>total += scores[scoreCount]</code> | <p>spelling errors as long as intention is clear.</p> <p>Allow description of change that would be made (e.g. "change 1 to 0")</p> <p>First correction is fixing indexing error so element 0 is included. This could be done on line 03 e.g. <code>scores[scoreCount-1]</code>. Second correction is fixing addition of total.</p> <p>If both errors fixed on line 03, full marks should be given. e.g. <code>total = total + scores[scoreCount-1]</code></p> <p><u>Examiner's Comments</u></p> <p>This proved to be a relatively challenging question for many candidates. This question relied on an understanding of the term "logic error".</p> <p>Two errors were present:</p> <ul style="list-style-type: none"> • line 02 (where the count-controlled loop ignored element 0 in the array) • line 03 (where the total was incorrectly calculated and stored). <p>Many responses identified at least one of the line numbers containing the error. Far fewer were able to successfully fix the errors satisfactorily.</p> <p>Examiners were instructed to be generous in interpreting potential fixes. The errors in either case could have been fixed using OCR Exam Reference Language (as presented) or using any other sensible form. Responses using programming syntax were credited, as were those who simply used</p> |
|--|--|---|---|

| | | | | |
|---|---|----|--|---|
| | | | | English, e.g. "change the 1 to a 0 on line 02". |
| | | | | Instructions were also included in the mark scheme to allow full marks for candidates who fixed the errors in one step on line 03. |
| | | | Total | 6 |
| 7 | a | i | <p>1 mark each to max 2</p> <ul style="list-style-type: none"> • Check the program works (as intended) • meets user requirements. • gives the correct output / result • Find / detect / check for errors / bugs • Check the program does not crash / is robust / executes / runs • To try and break the program / destructive testing • Test for / improve usability / user experience / performance / check user feedback is suitable • Allow any errors to be fixed / make changes / improvements as a result of testing • Ensure no problems / issues fixed when released. • Defensive design considerations / anticipating misuse / so cannot be misused | <p>2 (AO1 1b)</p> <p>Allow answers that explain what would happen if not tested (e.g. "there might be bugs")</p> <p><u>Examiner's Comments</u></p> <p>Testing as a process could be done for many reasons. The mark scheme attempts to credit as many sensible explanations for this as possible. This includes testing to find errors, checking for robustness and checking against user requirements. Furthermore, fixing errors that are found is also credited as this could form part of the testing process.</p> <p>The majority of responses demonstrated an understanding of this and achieved highly.</p> |
| | | ii | <p>1 mark for name, 1 mark for matching description</p> <p>e.g.</p> <ul style="list-style-type: none"> • Final / terminal testing... • ... Completed at the end of development / before release. • ... to test the product as a whole. • Iterative / incremental testing... • ...completed during development. • ...after each module is completed. • ... test module in isolation • Normal testing... • ...test using data that should be accepted / • ...test that is expected to work / pass • Boundary / Extreme testing... • ...test using data that is on the edge of being acceptable / unacceptable. • ...test highest / lowest value • Invalid / Erroneous testing... | <p>2 (1 AO1 1a) (1 AO2 1b)</p> <p>Allow other sensible descriptive names for testing.</p> <p>Description must match test type.</p> <p>Must be a description and not just an example, but example may support description.</p> <p>Do not accept descriptions that simply repeat type of test without further clarification (e.g. "boundary, testing the boundary").</p> <p>Allow :</p> <ul style="list-style-type: none"> • Black box testing... • ...testing without access / knowledge of a system's workings. |

| | | | |
|--|--|---|---|
| | | <ul style="list-style-type: none"> ...test using data that should be rejected / is not acceptable / causes an error | <ul style="list-style-type: none"> White box testing... ...testing with access / knowledge of system's workings. <p>Allow other sensible / valid types of testing.</p> <p>Do not accept examples of validation (e.g. type test, range check)</p> <p>"data that is not expected" is NE for invalid/erroneous unless clarified.</p> <p><u>Examiner's Comments</u></p> <p>The J277 specification lists iterative and final/terminal testing as test types. However, many candidates interpreted this question as asking about test data (such as normal or erroneous data). Where candidates described the use of test data and link it to expected outcomes, this was credited by examiners.</p> <p>Other types of suitable testing that do not appear on the J277 specification (such as white box/black box testing, alpha/beta testing) were also accepted.</p> <p> Assessment for learning</p> <p>The J277 specification states the minimum content that candidates are expected to know and understand at GCSE level. However, it is possible for teachers to go beyond this.</p> <p>For example, iterative and final/terminal testing are stated in the specification. However, there are other types of testing. Other technically correct responses will be accepted by examiners even if</p> |
|--|--|---|---|

| | | | | |
|--|-----|---|---------------------------|--|
| | | | | <p>they do not appear in this specification.</p> <p>Another example is sorting algorithms. Merge sort, bubble sort and insertion sort appear in the specification. However, when candidates have been asked to name sorting algorithms, previous mark schemes credit other valid responses (such as quick sort, selection sort or bogo sort).</p> |
| | iii | <p>1 mark for feature 1 mark for matching description e.g.</p> <ul style="list-style-type: none"> • Translator / compiler / interpreter ... • ... convert to low-level/machine code • ...allow program to be executed / run • ...produce executable file (only for compiler) • ...stops execution when error found (interpreter only) • Run-time environment / output window... • ...allows program / code to be run / executed • ...shows output of the program / code • Error reporting / diagnostics • ... identify location/detail of errors • ...suggests fixes • Debugger ... • ...find errors • Stepping ... • ... execute/run the program line by line • Variable watch... • ... see the contents/data held in variables • Break points ... • ... will allow the program to stop at a chosen / set position • Text/code editor... • ...allows program code to be written / entered / changed • ...allows errors to be fixed • Pretty printing / keyword highlighting... • ... allows keywords / variables to be coloured / identified • Keyword completion / syntax suggestion... • ...suggests code/syntax when first part entered. | <p>4 (AO2 1b)</p> | <p>Allow other sensible names for features.</p> <p>Description must add more than is given in the identification of the feature to be awarded. For example, “keyword highlighting, highlights keywords” is 1 mark for the feature only.</p> <p>If compiler and interpreter given as two distinct features, allow both (with suitable descriptions). Do not allow translator and compiler/interpreter.</p> <p>Description must match feature.</p> <p>“finding errors” is NE for description of error reporting.</p> <p>Allow sensible references to AI where appropriate. Sensible description of use needed.</p> <p>Allow other sensible features of an IDE (e.g. line numbering, auto indent, collapsed blocks, etc) with suitable description.</p> <p>For text editor / error diagnostics / debugger, allow other sensible features listed as features in the mark scheme as description (e.g. “text editor, provides pretty printing”, “debugger, provides stepping”)</p> |

| | | | | |
|--|---|--|--|--|
| | | | | <p><u>Examiner's Comments</u></p> <p>This question was generally well answered with a variety of features given. The question specifically asks about features used when testing a program. Therefore, features such as debugging tools, stepping and variable watch windows were very common responses.</p> <p>However, more general responses were also accepted, such as text editors, translators, and keyword completion. These could all potentially be used when editing programs after errors had been identified.</p> <p>Less successful responses tended to be descriptions that simply repeated the name of the feature given. For example "debugging tools, to allow debugging" would gain 1 mark for the feature identification but not the description.</p> |
| | b | <p>1 mark for method, 1 mark to max 2 for each use</p> <p>e.g.</p> <ul style="list-style-type: none"> • Range check • ... checks upper/max / lower/min / boundaries • ... make sure the players answer / input is between sensible limits (e.g. 20 or less, between 2 and 20 inclusive) / not negative • ...by example of program code • Type check • ... making sure the data inputted is of the correct data type • ... make sure answer / input is an integer (or equivalent e.g. whole number) • Presence check • ... making sure a value is inputted / not blank • ... reference to answer / input • ...by example of program code • Length check | <p>6 (4 AO2 1b) (2 AO1 1a)</p> | <p>Validation must be applied to the rules of the game as given; do not accept uses related to input not asked for (e.g. names, passwords, etc).</p> <p>Do not accept uses that simply repeat the name of the check (e.g. "range check, checks a range of numbers")</p> <p>For range check, values must be sensible (e.g. 1 to 50), and allow input of 2 to 20. Do not allow 1 / 10 (answer could be over this).</p> <p>For length check, must be clear that the string version of the data input is being checked to award use marks (e.g. characters not digits).</p> |

| | | | | |
|--|--|--|--|---|
| | | | <ul style="list-style-type: none"> ... limit number of characters / check maximum / minimum string length ... answer / input must be 1 or 2 characters ...by example of program code Format check ... making sure the data inputted follows a set pattern ... checking answer / input consists of only 1 or 2 numeric digits ...by example of program code Look up / table check ... making sure the data inputted is one from an allowed set of values ... checking that answer / input is one of [2, 3...20] inclusive ...by example of program code | <p>Accept alternative names or descriptions (e.g. existence check, boundary check) but name of check must be sensible.</p> <p>Mark each answer as a whole, ignore method/use headings.</p> <p>Do not accept defensive design elements (e.g. input sanitisation, authentication)</p> <p>Examples of program code can be actual code (e.g. <code>if inp>=2 and inp<=20</code>) or identification of technique (e.g. "use IF statement / Case statement to limit values to between 1 and 20"). Do not accept code just showing casting.</p> <p><u>Examiner's Comments</u></p> <p>Responses which focused on the explicit link to the game described in this question tended to do well.</p> <p>The stronger responses stated a validation method and linked the use of the validation method to the game. For example, a requirement of the game is that two random numbers between 1 and 10 are picked. It is sensible to suggest that validation ensuring the total is between 2 and 20 could be implemented. Further discussion relating to how this could be done, even as far as suggesting sensible high-level code that could be used would have developed the response.</p> <p>Some candidates gave examples of validation which did not clearly link to the game. Generic examples were partially credited. Explicit links to the game were required in order to gain all marks available.</p> |
|--|--|--|--|---|


| | | | | | | | |
|---|--|---|--|--|--|---------------------------------|--|
| | | | | | | | <div><div>?</div><div>Misconception</div></div> <p>Input validation applies to values input by the user. In this case, the requested input is the sum of two numbers, each of which are between 1 and 10. It is not necessary to validate the random number generation (as this has not been input) and it would be inappropriate to limit user inputs to between 1 and 10; the total could easily be (for example) 8 + 6 = 14.</p> <p>Where candidates suggested validating inputs to allow between 1 and 10, not all marks available were given due to this misconception.</p> |
| | | | Total | | | 14 | |
| 8 | | | 1 mark for each row | | | 4 (AO3 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> |
| | | | | | | | |
| | | | Total | | | 4 | |
| 9 | | i | <ul style="list-style-type: none">Convert/change one data type to anotherLine 03 / 3 / three | | | 2 (AO1 1b) (AO2 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</p> |

| | | | | <p>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 03 • Kofi2021x as staffID on line 05 • Kofi2021xx as staffID on line 05 • ID Kofi2021xx output on line 07 as first and only output | <p>4 (AO3 2c)</p> | <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 border="1"> <thead> <tr> <th>Line number</th><th>surname</th><th>year</th><th>staffID</th><th>Output</th></tr> </thead> <tbody> <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> <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> </tbody> </table> <p><u>Examiner's Comments</u></p> <p>This question asked candidates to trace through a given algorithm to</p> | Line number | surname | year | staffID | Output | 01 | Kofi | | | | 02 | | 2021 | | | 03 | | | Kofi2021 | | 05 | | | Kofi2021x | | 05 | | | Kofi2021xx | | 07 | | | | ID Kofi2021xx |
| Line number | surname | year | staffID | Output | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | Kofi | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 02 | | 2021 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 03 | | | Kofi2021 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 05 | | | Kofi2021x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 05 | | | Kofi2021xx | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 07 | | | | ID Kofi2021xx | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | |
|----|--|--|--|--|
| | | | | <p>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 |
| 10 | | | <p>Any two bullet points for one mark each:</p> <ul style="list-style-type: none"> • Add comments • Name variables sensibly • Put into subroutine / procedure / function • Use loop / iteration | <p>2 (AO2 1b)</p> <p>Do not accept indentation (no code to sensibly indent in this example)</p> <p>“Use a subroutine” is not enough. Must be clear that existing code will be put into a new subroutine.</p> <p><u>Examiner’s Comments</u></p> <p>This question asked about</p> |

| | | | | |
|----|---|--|------------------|--|
| | | | | <p>maintainability. It was important that a candidate's response referred to the code given.</p> <p>"Give two ways that maintainability of a program could be improved" is a different question than the one asked.</p> <p>Because of this, responses that were accepted must genuinely improve the maintainability of the program shown. One very common wrong response was indentation; although this would be useful generically, there was no code presented that would benefit from being indented.</p> <p>Many candidates were able to give responses such as use of comments and sensibly named variables that would genuinely improve the given program. Where candidates described putting the given code inside a newly defined subroutine. This response was credited. However some responses simply said "use a subroutine". This was not enough.</p> |
| | | | Total | 2 |
| 11 | 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) ...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</i></p> | 5 (AO3 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> <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> |

| | | | |
|--|--|--|--|
| | | <p><i>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> | <pre> valid = True if firstname == "" or surname == "" then valid = False end if if room != "basic" and room != "premium" then valid = False endif if nights < 1 or 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"> • if firstname != "" and surname != "" • if room == "basic" or room == "premium" • if nights >= 1 and nights <= 5 <p><u>Examiner's Comments</u></p> <p>This question stretched the understanding of even highly-achieving candidates and it was not uncommon to see low scoring responses.</p> <p>Misunderstanding of Boolean operators (AND and OR) within selection (IF) statements was something that affected a lot of candidate responses.</p> |
|--|--|--|--|

| | | | | | |
|--|--|--|--|--|--|
| | | | | | <p>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.</p> <p>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.</p> <p> Misconception</p> <p>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 :</p> <ul style="list-style-type: none"> • <code>IF room == "basic" or "premium"</code> is incorrect as the second part of the statement is not evaluated against anything. This was perhaps the most common mistake. • <code>IF room == "basic" or room == "premium"</code> is correct and checks for validity. • <code>IF room == basic or room == premium</code> is incorrect as the lack of string delimiters means that basic and room would be treated as variables rather than strings. |
|--|--|--|--|--|--|

| | | | | |
|--|--|--|--|---|
| | | | | <div><ul style="list-style-type: none">• 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.</div> <div>The same explanation follows for the other two necessary checks.</div> <div>Exemplar 3</div> <div><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</pre></div> <div>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 and the program ends. Efficient use of if ... else ... means that the next check only proceeds if the previous check returns False.</div> <div>If all three checks return False, the</div> |
|--|--|--|--|---|

| | | | | | <p>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 (AO3 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 | | | | | | | | | | | | | | | |
| | | | Total | 8 | | | | | | | | | | | | | |
| 12 | a | i | <ul style="list-style-type: none">• Hiding / ignoring / removing detail / focussing on certain parts of a problem | 1 | | | | | | | | | | | | | |
| | | ii | <ul style="list-style-type: none">• Focus on age / number of miles• Ignore other factors (such as make, model, etc) | 1 | Allow other examples of factors to ignore / remove for BP2 | | | | | | | | | | | | |
| | | iii | <ul style="list-style-type: none">• Ensures only certain users can access the system• Using password / other example of authentication technique | 2 | Allow other examples of authentication for BP2 | | | | | | | | | | | | |
| | b | i | 1 mark per bullet, max 4 <ul style="list-style-type: none">• Miles and age input <u>separately</u> | 5 | BP2 and 3 must check for both ends of range – must check that input data is not negative. | | | | | | | | | | | | |

| | | | <ul style="list-style-type: none">Checks for valid mileageChecks for valid ageChecks <u>both</u> are greater than / greater than equal to zero...correctly outputs both True and False | | <p>Allow FT for BP4 if already penalised under BP2 and/or 3 and output is otherwise correct.</p> <p>e.g.</p> <pre>miles = input("enter miles driven") age = input("enter age of car") valid = True if miles > 10000 or miles < 0 then valid = False elseif age > 5 or age < 0 then valid = False endif print(valid)</pre> | | | | | | | | | | | | | | | |
|-------------|-----------|-----------------------|---|-------------|---|-----------------------|-------|--------|--------|-------|----------|-----------|--------|--------|----------|--------|----------|-----|---|--|
| | | ii | <p>1 mark per row, max 3</p> <ul style="list-style-type: none">Normal : miles (0 – 9,999), age (0 - 5)Erroneous: miles (less than 0, larger than 9,999), age (less than 0 / more than 5) / non-numeric dataBoundary : miles (-1/0 / 9,999 / 10,000), age (-1/0 / 5/6) | 3 | <p>Specific data must be given, not a description</p> <p>e.g.</p> <table><tr><td></td><td>Miles</td><td>Age</td></tr><tr><td>Normal</td><td>7,000</td><td>3</td></tr><tr><td>Erroneous</td><td>12,000</td><td>7</td></tr><tr><td>Boundary</td><td>10,000</td><td>5</td></tr></table> | | Miles | Age | Normal | 7,000 | 3 | Erroneous | 12,000 | 7 | Boundary | 10,000 | 5 | | | |
| | Miles | Age | | | | | | | | | | | | | | | | | | |
| Normal | 7,000 | 3 | | | | | | | | | | | | | | | | | | |
| Erroneous | 12,000 | 7 | | | | | | | | | | | | | | | | | | |
| Boundary | 10,000 | 5 | | | | | | | | | | | | | | | | | | |
| | | iii | <ul style="list-style-type: none">During development / whilst writing the program / before development is complete. | 1 | | | | | | | | | | | | | | | | |
| | | | Total | 13 | | | | | | | | | | | | | | | | |
| 13 | a | i | <p>One mark if two correct, two marks if four correct, three marks if all correct.</p> <table><tr><th>Price input</th><th>Test type</th><th>Expected price output</th></tr><tr><td>50</td><td>Normal</td><td>50</td></tr><tr><td>100</td><td>Boundary</td><td>100</td></tr><tr><td>150</td><td>Normal</td><td>130</td></tr><tr><td>200</td><td>Boundary</td><td>180</td></tr></table> | Price input | Test type | Expected price output | 50 | Normal | 50 | 100 | Boundary | 100 | 150 | Normal | 130 | 200 | Boundary | 180 | 3 | |
| Price input | Test type | Expected price output | | | | | | | | | | | | | | | | | | |
| 50 | Normal | 50 | | | | | | | | | | | | | | | | | | |
| 100 | Boundary | 100 | | | | | | | | | | | | | | | | | | |
| 150 | Normal | 130 | | | | | | | | | | | | | | | | | | |
| 200 | Boundary | 180 | | | | | | | | | | | | | | | | | | |

| | | | | | | | |
|--|--|----|--|--------|-----|---|---|
| | | | 250 | Normal | 210 | | |
| | | ii | <p>One mark per bullet point</p> <ul style="list-style-type: none"> • Input and store price • Check if price is > 200... • ...if true, reduce price by 40 • Check if price is >100 <u>and not >200</u>... • ...if true, reduce price by 20 • Output price | | | 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 and BP5 only to be given if sensible check for price being over the appropriate threshold. BP4 must check that price is both larger than 100 and not larger than 200; do not give mark for simply checking price is larger than 100. This may be implicit (e.g. using elseif).</p> <p>e.g.</p> <pre>price = input("enter price") if price > 200 then price = price - 40 elseif price > 100 then price = price - 20 endif print(price)</pre> |
| | | b | <p>One mark per bullet point</p> <ul style="list-style-type: none"> • checking both values (e.g. or changed to and if appropriate) • if statement in correct format (e.g. checking against stocklevel for each condition) • if statement uses correct comparisons (e.g. >= and <=) • print statements in correct position • print statements include string delimiters (e.g. speech marks) around both string outputs | | | 5 | <p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural language.</p> <p>e.g.</p> <pre>stocklevel = input("Enter stock level") if stocklevel >= 5 and stocklevel <= 25 then print("In demand") else print("Not in demand") endif</pre> <p>alternative example</p> <pre>stocklevel = input("Enter stock level")</pre> |

| | | | | | |
|----|---|----|--|---------------|--|
| | | | | | <pre>if stocklevel > 5 or stocklevel > 25 then print("Not in demand") else print("In demand") endif</pre> <p>As a matter of principle, a candidate who refines the program to work fully but in a different format to that specified should gain full marks.</p> |
| | | | Total | 14 | |
| 14 | a | i | <ul style="list-style-type: none">• or• >300 / >= 301• print | 3 (AO3 2b) | <p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural English.</p> <p>MP2 do not accept 'greater than', must use the HLL syntax > or >= MP3 must be a suitable output command word that could be found in a HLL e.g. print (Python), console.writeline (VB), cout (C++)</p> |
| | | ii | <ul style="list-style-type: none">• Suitable invalid test data (i.e. > 300, e.g. 350)• Suitable boundary test data (e.g. 0, 300)• "Value accepted" or equivalent if boundary data 0 or 300 / "Invalid input displayed" or equivalent if boundary data -1 or 301 | 3 (AO3 2c) | |
| | b | | <ul style="list-style-type: none">• Initialises total as 0 and prints out total the end (as per original program)• Uses iteration, e.g. FOR, WHILE• ...that repeats 5 times• ...correctly adds up values using loop index <p>e.g.</p> <pre>total = 0 for x = 0 to 4 total = total + hoursplayed[2, x] next x</pre> | 4 (AO3 2c) | <p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural English.</p> <p>MP1 must have appropriate identifier, = and then the numeric 0 MP2 must have for or while MP3 must have the for stopping condition 4/5 MP4 must have the same identifier</p> |

| | | | | |
|--|--|--|-----------|--|
| | | <pre>console.writeline(total)</pre> <p>e.g.</p> <pre>total = 0 for x in range (0, 4) total += hoursplayed[2][x] next x print (total)</pre> <p>e.g.</p> <pre>total = 0; for (int x = 0; x <= 4; x++){ total = total + hoursplayed[2][x]; } System.out.println (total);</pre> | | <p>for MP1 and equal and + to add the data in the array (using either [x,y] or [x][y]. This could be total = total + Or total +=</p> |
| | | Total | 10 | |