Chhetri Academy

chhetriacademy.com

GCSE COMPUTER SCIENCE

2.1

OCR J277 | Specification point 2.1.2

# Designing, creating and refining algorithms

## Untiered

Answers and examiner comments

| 5 | 2018-2025 | Untiered |
| --- | --- | --- |
| question groups | available papers | tier |

### Using this booklet

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

**Name** _____ **Date** _____

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

| | | | | | |
|---|---|---|---|---|---|
| **2** | **(a)** | **(i)** | • 2, 3, 4 | 1 | All three numbers needed in the correct order (with no other numbers) for mark. |
| **2** | **(a)** | **(ii)** | • 15 | 1 | Accept 3 x 5 |
| **2** | **(b)** | | 1 mark per bullet, max 2<br><br>• Sequence<br>• Iteration / loops / repetition | 2 | Ignore spelling.<br><br>Do not allow examples (eg FOR loop / WHILE loop) |
| **2** | **(c)** | **(i)** | 1 mark per bullet, max 2<br><br>• A (name/identifier for a) **memory location**<br>• used to (temporarily) **holds/contains/stores** data / value // is assigned a value<br>• that **can be** changed / **possible** to change (while the program is running) | 2 | Do not accept "will change" for bullet point 4.<br><br>Do not allow "holds/stores something" or "holds/stores information" for bullet point 2.<br><br>Do not accept name / identifier without reference to a memory location. Do not accept "a value given a name" or equivalent. |
| **2** | **(c)** | **(ii)** | 1 mark per bullet, max 2<br><br>• k<br>• p<br>• m | 2 | Ignore capitalisation.<br><br>Correct answer only. Do not allow other code in answer. |

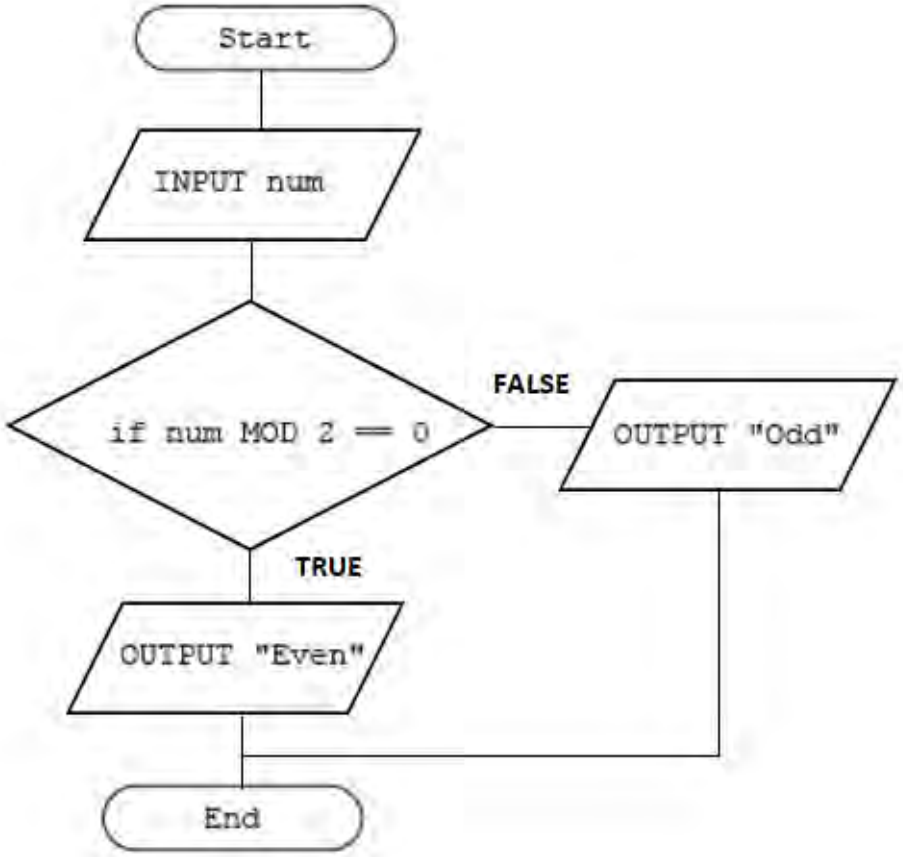
# Original question 2

The full source mark scheme begins on page 22. Use the original paper code, tier and question number shown above.

<table>
<tr><th colspan="3">Question</th><th>Answer</th><th>Mark</th><th>Guidance</th></tr>
<tr><td>1</td><td>(a)</td><td></td><td>1 mark per correct row<br><br>
<table>
<tr><th>OCR Reference Language code</th><th>Selection</th><th>Iteration</th></tr>
<tr><td><pre>for i = 1 to 10
    print(i)
next i</pre></td><td></td><td>✓</td></tr>
<tr><td><pre>whilescore != 0
    playgame()
endwhile</pre></td><td></td><td>✓</td></tr>
<tr><td><pre>if playerHit() then
    score = score - 1
endif</pre></td><td>✓</td><td></td></tr>
<tr><td><pre>switch bonus:
    case 0:
        score = 9
    case 1:
        score = 7
    case 2:
        score = 5
endswitch</pre></td><td>✓</td><td></td></tr>
</table></td><td>4<br>(AO2 1b)</td><td>No mark given if both boxes in a row ticked.<br><br>Accept any response (ticks, crosses, etc) that clearly indicates candidate’s choice.</td></tr>
<tr><td>1</td><td>(b)</td><td></td><td>• <code>score = score + 1 // score +=1 // score++</code></td><td>1<br>(AO3 2b)</td><td>Allow other logically correct answers that result in score increasing by one and being overwritten. Do not accept <code>score + 1 / score = +1</code><br><br>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”<br><br>Ignore any superfluous code that does not affect the outcon</td></tr>
<tr><td></td><td>(c)</td><td></td><td>• Decomposition</td><td></td><td>Correct answer only. Ignore spelling errors.</td></tr>
</table>

| | | | | | |
|---|---|---|---|---|---|
| | | | • Abstraction | **(AO1 1a)** | |

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

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

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

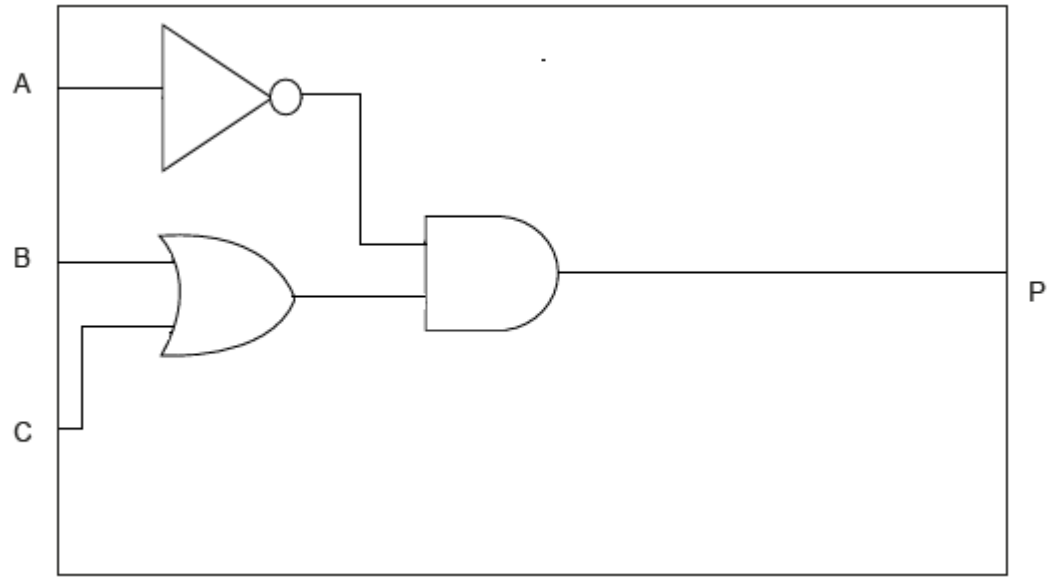
| | | | | | |
|---|---|---|---|---|---|
| 3 | (b) | | 1 mark each<br>• line 03<br>• `total = num1 + num2`<br><br>• Line 04<br>• `if total >= 10 and total <=20 then`<br><br>Allow other logical equivalent code e.g.<br>`total = int(num1) + int(num2)`<br>`if 10 <= total <= 20` | 4 (AO3) | Allow other logical corrections that will fix the problem identified and does not introduce any further errors.<br><br>Allow descriptions of changes as long as clear exactly what will change. Do not allow ambiguous descriptions of changes to code.<br><br>Ignore missing `then` from line 04.<br><br>Ignore capitalisation. |
| 3 | (c) | (i) | 1 mark each<br>• Compare to / pick out middle value (which is 6)<br>• **discard only left side // retain only right side** (because 6 < 10)…<br>• …Compare to / pick out (middle value which is) **10** | 3 (AO2) | BP1 can be given for generic answer. BP2 and 3 must be linked to data set given<br><br>For BP2, must remove 1, 2,5 and 6 from list if discussing individual numbers. Allow FT for BP3 if this done incorrectly. |

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

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

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

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

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

<table>
<tr>
<td>6</td>
<td>(a)</td>
<td></td>
<td>1 mark for each output
<table>
<tr><td>print(message.upper)</td><td>ABCD1234 (upper case)</td></tr>
<tr><td>print(message.left(4))</td><td>abcd (lower case)</td></tr>
<tr><td>print(int(message.right(4))*2)</td><td>2468</td></tr>
</table></td>
<td>3<br>(AO2)</td>
<td>Case must be correct but BOD if ambiguous.<br><br>Allow quotation marks in answer.</td>
</tr>
<tr>
<td>6</td>
<td>(b)</td>
<td></td>
<td>1 mark per bullet point :<br>
• storing both strings correctly in <code>word1</code> and <code>word2</code><br>
• correct concatenation (<code>word1</code> then <code>word2</code>)...<br>
• …storing in variable <code>message</code><br><br>
Example<br>
<pre>word1 = "Hello"
word2 = "Everyone"
message = word1 + word2</pre></td>
<td>3<br>(AO3)</td>
<td>Accept & / + / . etc as valid methods of concatenation. Allow use of sensible concatenation functions e.g. concat() . Do not allow commas.<br><br>Do not allow == for assigning value to string. Do not allow spaces in variable names. Penalise once then FT.<br><br>Ignore additional code given. Ignore case.<br><br>Reasonable attempt at BP2 needed to access BP3.</td>
</tr>
</table>

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

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

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

## Section B

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

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

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

*PMT*

# GCSE (9–1)

# Computer Science

**J276/02:** Computational thinking, algorithms and programming

General Certificate of Secondary Education

## Mark Scheme for June 2019

*PMT*

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

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

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

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

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

© OCR 2019

*PMT*

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

## Annotations

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

PMT

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

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

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

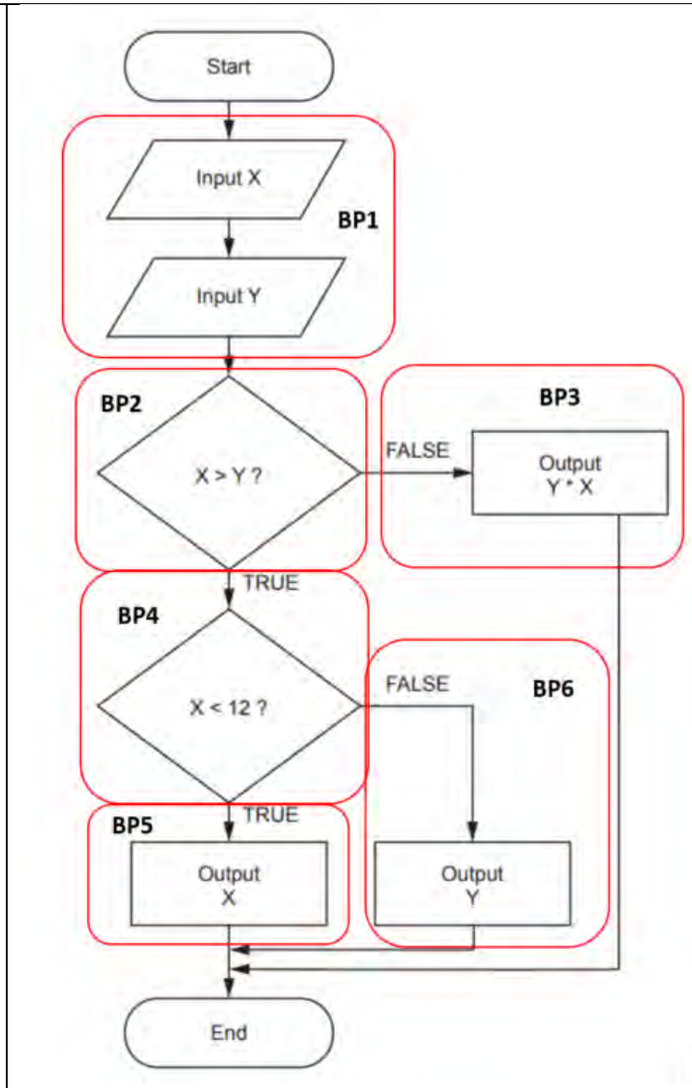
| | | | | | |
|---|---|---|---|---|---|
| **1** | **(b)** | **(ii)** | 1 mark per bullet to max 2.<br><br>• Computers consist of transistors / switches / logic circuits / gates…<br>• …which only have two values / on or off / 1 or 0 / open or closed | 2<br>AO1 1b (2) | Only give BP2 if BP1 given.<br>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

*PMT*

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

| | | | | | |
|---|---|---|---|---|---|
| **2** | **(a)** | | 1 mark per bullet to max 4, 1 mark per row<br><br>• 10<br>• 6<br>• 6<br>• 2 | 4<br>AO2 1b (4) | Correct Answer Only<br><br>Do not accept “X”, “Y”, etc. |
| **2** | **(b)** | | 1 mark per bullet to max 6.<br><br>• Inputs two value (as X and Y)<br>• Compares if X is larger than Y…<br>• …Outputs Y*X only when <u>False</u><br>• Compares if X is less than 12…<br>• …Outputs X only when True <u>and X > Y</u><br>• …Outputs Y only when False <u>and X > Y</u> | 6<br>AO3 2b (6) | Question specifically asks for pseudocode.<br><br>Outputs should only be given if they occur with the right condition(s).<br><br><u>Example algorithm</u><br><br>`input x`<br>`input y`<br>`if x > y then`<br>`  if  x < 12 then`<br>`  print x`<br>`  else`<br>`  print y`<br>`  end if`<br>`else`<br>`  print y*x`<br>`end if`<br><br>Variables do not have to be called x and y.<br><br>Accept equivalent comparisons (e.g. `if X <= Y`)<br><br>Allow FT for outputs from incorrect comparisons where a sensible attempt has been made. |

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

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

5

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

| | | | | | |
|---|---|---|---|---|---|
| **2** | **(c)** | | 1 mark per bullet to max 4, 2 mark max per method<br><br>• Compiler<br>• …translates code **in one go** / all at once<br>• …produces an executable file // does not need to be compiled again<br><br>• Interpreter<br>• …translates code line by line.<br>• …will be interpreted / translated every time it is run. | 4<br>AO1 1b (4) | Mark first method only in each section |
| **3** | **(a)** | **(i)** | 1 mark per bullet to max 1<br><br>• An error that results in **incorrect output / unexpected result**<br>• Contains an error but **still runs / doesn't crash** | 1<br>AO1 1b (1) | Do not accept examples of logic errors. |
| **3** | **(a)** | **(ii)** | `if num MOD 2 == 0 then`<br><br>`if num MOD 2 = 0 then` | 1<br>AO3 2b (1) | Important point is that >= is changed to == or =.<br>Accept alternatives that produce the same result (e.g. <=0, <1, !=1, etc.)<br><br>Ignore any casting (e.g. using int() to convert to a number)<br><br>Accept other minor changes to the line as long it logically works.<br><br>Accept versions of MOD from high level languages (e.g. Python : `if num % 2 == 0` ) |

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

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

*PMT*

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

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

8

*PMT*

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

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

*PMT*

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

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

10

*PMT*

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

| | | | | | |
|---|---|---|---|---|---|
| **6** | **(a)** | **(i)** | 1 mark per bullet to max 6<br><br>• Function ticketprice() **defined**<br>• … that accepts two parameters and has no other inputs<br>• Works out total ticket price for adult (eg adult * 19.99)<br>• Works out total ticket price for children (eg child * 8.99)<br>• Adds on **correct** booking fee<br>• Returns the calculated value. | 6<br>AO3 2b (6) | Bullet points 3, 4, 5 can be awarded even if no mention of a function / parameters (for example, if candidate has inputted the number of tickets needed.<br><br>**Do not** award return value if no attempt at a function. Return mark can be given if a good attempt made at calculating the total, even if this is incorrect.<br><br>Allow 2.50 booking fee to be per order or per ticket<br><br>Ticket prices must be stored appropriately if needed.<br><br>example algorithm<br><br>`function ticketprice(numadult, numchild)`<br>`  price = (numadult * 19.99) + (numchild * 8.99) + 2.50`<br>`  return price`<br>`end function`<br><br>Allow alternatives in high level languages (e.g. def in Python).<br><br>Allow return as assigning the value to the name of the function (VB syntax) |
| **6** | **(a)** | **(ii)** | 1 mark per bullet to max 2<br><br>• Real…<br>• …Returned value may not be a whole number / may have a decimal point in | 2<br>AO2 1a (1)<br>AO2 1b (1) | Allow String only if matching justification shows understanding (e.g. £ sign attached, message returned alongside value). |

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

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

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

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

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

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

*PMT*

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

**OCR Customer Contact Centre**

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

**www.ocr.org.uk**

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

**Oxford Cambridge and RSA Examinations**
**is a Company Limited by Guarantee**
**Registered in England**
**Registered Office; The Triangle Building, Shaftesbury Road, Cambridge, CB2 8EA**
**Registered Company Number: 3484466**
**OCR is an exempt Charity**

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

**© OCR 2019**

## Question 1 (a)

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

| OCR Reference Language code | Selection | Iteration |
|---|---|---|
| `for i = 1 to 10`<br>`  print(i)`<br>`next i` | | |
| `while score != 0`<br>`  playgame()`<br>`endwhile` | | |
| `if playerHit() then`<br>`  score = 0`<br>`endif` | | |
| `switch bonus:`<br>`  case 0:`<br>`  score = 9`<br>`  case 1:`<br>`  score = 7`<br>`  case 2:`<br>`  score = 5`<br>`endswitch` | | |

**[4]**

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

## Question 1 (b)

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

.....

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

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

### Assessment for learning

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

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

## Question 1 (c)

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

Breaking a complex problem down into smaller problems.

.....

.....

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

.....

.....

**[2]**

This question was answered well by the majority of candidates.

## Question 1

**1** Tick (✓) **one** box in each row to identify the programming construct where each keyword is used.

| Keyword | Programming construct | |
|---|---|---|
| | Selection | Iteration |
| `if` | | |
| `for` | | |
| `while` | | |

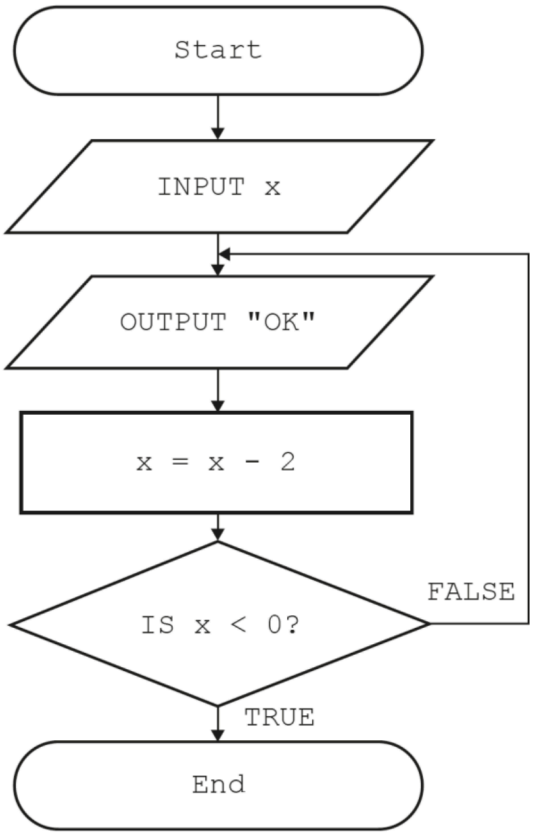
**[3]**

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.

Practical programming skills in lessons should be used to make this explicit link between technical definitions and use within a high-level language.

## Question 2 (a)

**2** This flowchart uses iteration.




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

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

**[2]**

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

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

## Question 2 (b) (i)

**(b)** Iteration is a programming construct.

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

1 .....

.....

.....

2 .....

.....

.....

**[4]**

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

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

### Misconception

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

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

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

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

## Question 2 (b) (ii)

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

1 .....

2 .....

**[2]**

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

## Question 2 (c)

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

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

Tool 1 .....

Description .....

.....

Tool 2 .....

Description .....

.....

**[4]**

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

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

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