Cambridge Assessment International Education

# Cambridge International AS & A Level

**COMPUTER SCIENCE** **9618/32**

Paper 3 Advanced Theory **May/June 2021**

MARK SCHEME

Maximum Mark: 75

**Published**

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the May/June 2021 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.





**[Turn over**

## Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

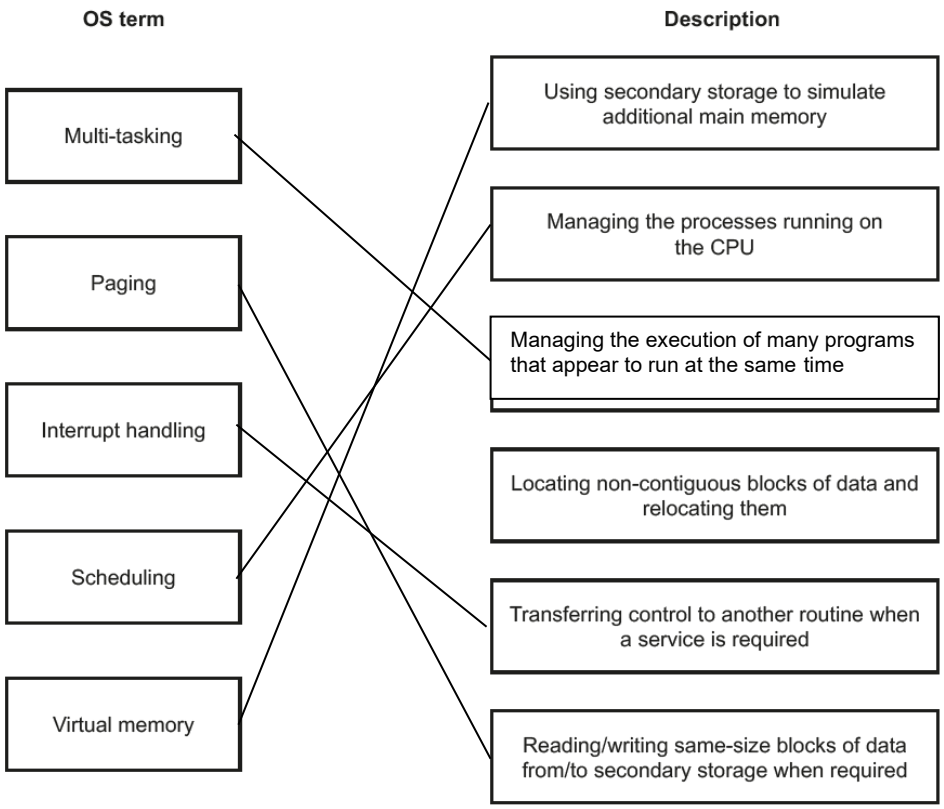
GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

| Question | Answer | Marks |
|---|---|---|
| 1(a) | Working: **one mark** for calculation of the mantissa and **one mark** for calculation or use of the exponent<br><br>**Exponent: one** from:<br>= $0.11101 \times 2^3$ // $0.11101 \times 2^{11}$ // $0.11101 \times 10^3$ // $0.11101 \times 10^{11}$<br>= $1.00011 \times 2^3$ // $1.00011 \times 2^{11}$ // $1.00011 \times 10^3$ // $1.00011 \times 10^{11}$<br>= appropriate shifting of binary point for +7.25<br><br>**Mantissa: one** from:<br>= 111.01 (conversion to binary +7.25 – 10 bits)<br>= 0111010000 (mantissa 10 bits for +7.25<br>= 1000101111(one's complement mantissa for –7.25)<br>= 1000110000 (two's complement mantissa for –7.25)<br><br>Correct Answer **(Max 1)**<br><br>Mantissa: 1 0 0 0 1 1 0 0 0 0 Exponent: 0 0 0 0 1 1 | **3** |
| 1(b) | **One** mark for working out the exponent<br>**One** mark for working out the mantissa<br>**One** mark for the correct answer<br><br>Example answers<br>• =$1.011000111 \times 2^7$ (exponent is 7)<br>• =10110001.11 // –128 + 32 + 16 + 1 + 0.5 + 0.25 // convert to positive 01001110.01 (and add a minus sign to the answer)<br>• –78.25 | **3** |
| 1(c) | **One** mark for working<br>**One** mark for correct mantissa<br>**One** mark for correct exponent<br><br>Example answers<br>Number of places added to exponent for normalisation −6 for number to retain its value // mantissa moved 6 places left<br><br>Mantissa: 0 1 1 1 0 0 0 0 0 0<br><br>Exponent: 1 0 0 0 0 1 | **3** |
| 1(d)(i) | **One** mark for each correct marking point (**Max 3**)<br><br>• Requires 11 bits / more than 10 bits to store (accurately) / reference to maximum (positive) number that can be stored = 511<br>• Denary 513 in binary is 1000000001 // Normalised: 0.1000000001<br>• Results in overflow | **3** |

| Question | Answer | Marks |
|---|---|---|
| 1(d)(ii) | **One** mark for each correct marking point (**Max 2)**<br><br>• The number of bits for the mantissa must be increased<br>• 11/12 bits mantissa **and** 5/4 bits exponent | **2** |

| Question | Answer | Marks |
|---|---|---|
| 2(a) | **One** mark for each correct marking point **(Max 2)**<br><br>• To create a new data type (from existing data types)<br>• To allow data types not available in a programming language to be constructed // To extend the flexibility of the programming language | **2** |
| 2(b)(i) | `TYPE SchoolDay = (Monday, Tuesday, Wednesday, Thursday, Friday)` | **1** |
| 2(b)(ii) | `TYPE WeekEnd = (Saturday, Sunday)` | **1** |
| 2(c) | **One** mark for each marking point (**Max 4**)<br><br>• `TYPE ClubMeet` and `ENDTYPE` correct<br>• `DECLARE FirstName` and `DECLARE LastName` included with correct data types<br>• `DECLARE Schoolday` included with correct data types from part 2(b)(i)<br>• `DECLARE Weekend` included with correct data types from part 2(b)(ii)<br><br>**Example answer**<br><br>`TYPE ClubMeet`<br>`  DECLARE FirstName : STRING`<br>`  DECLARE LastName : STRING`<br>`  DECLARE Schoolday : SchoolDay`<br>`  DECLARE Weekend : WeekEnd`<br>`ENDTYPE` | **4** |

| Question | Answer | Marks |
|---|---|---|
| 3(a) | **One** mark for each correct line from Operating System Term to Description<br>  <br> | **5** |
| 3(b) | **One** mark for each correct statement **(Max 4)**<br>• An interpreter examines source code one statement at a time<br>• Check each statement for errors<br>• …If no error is found the statement is executed<br>• …If an error is found this is reported and the interpreter halts<br>• Interpretation is repeated for every iteration in repeated sections of code/in loops<br>• Interpretation has to be repeated every time the program is run | **4** |

| Question | Answer | Marks |
|---|---|---|
| 4(a)(i) | **One** mark for each correct marking point **(Max 2)**<br>• Reverse Polish Notation provides an unambiguous method of representing an expression<br>• … reading from left to right<br>• …without the need to use brackets<br>• …with no need for rules of precedence / BODMAS | **2** |

| Question | Answer | Marks |
|---|---|---|
| 4(a)(ii) | **One mark** for identification of the data structure,<br>**One mark** for a sensible reason<br><br>**Either:**<br>Structure: stack<br>The operands are popped from the stack in the reverse order to how they were pushed<br><br>**Or:**<br>Structure: Binary tree<br>A (binary) tree allows both infix and postfix to be evaluated (tree traversal) | **2** |
| 4(b) | `a b - a c + * 7 /` | **1** |
| 4(c) | `a / b * 4 – (a + b)` | **1** |
| 4(d) | **1 mark** for correct structure<br>**1 mark** for correct substitution<br><br>`(a + b) / (c / d)`<br>`(17 + 3) / (48 / 12)` | **2** |

| Question | Answer | Marks |
|---|---|---|
| 5(a) | Working **(Max 3)**<br><br>May be seen on diagram<br>• Initialisation: setting Base to 0<br>• … and the rest of the towns to ∞<br>• Evidence to show values at nodes being updated<br>• Evidence to show 'visited node(s)'<br><br>May be seen in working section of paper<br>• Evidence to show calculation of at least one route<br>• Evidence to show more than one route has been calculated for at least one town<br><br>**Correct Answer (Max 2)**<br>**One mark** for four correct values…<br>… **One mark** for all values correct | **5** |

| Town 1 | Town 2 | Town 3 | Town 4 | Town 5 | Town 6 |
|---|---|---|---|---|---|
| 3 | 5 | 2 | 9 | 3 | 8 |

| Question | Answer | Marks |
|---|---|---|
| 5(b) | **One** mark for each correct marking point **(Max 3)**<br><br>• Artificial Neural Networks can be represented using graphs<br>• Graphs provide structures for relationships // graphs provide relationships between nodes<br>• AI problems can be defined/solved as finding a path in a graph<br>• Graphs may be analysed/ingested by a range of algorithms<br>• …e.g. A* / Dijksta's algorithm<br>• …used in machine learning.<br>• Example of method e.g. Back propagation of errors / regression methods | **3** |

| Question | Answer | Marks |
|---|---|---|
| 6 | **One** mark for each correct benefit **(Max 2)**<br>• Accuracy – Ensures accurate delivery of the message<br>• Completeness – Missing packets can be easily detected and a re-send request sent so the message arrives complete<br>• Resilience – if a network changes the router can detect this and send the data another way to ensure it arrives<br>• Path also available to other users // Doesn't use whole bandwidth // allows simultaneous use of channel by multiple users<br>• Better security as packets hashed and sent by different routes.<br><br>**One** mark for each correct drawback **(Max 2)**<br>• Time delays to correct errors // Network problems may introduce errors in packets<br>• Requires complex protocols for delivery<br>• Unsuitable for real time transmission applications | **4** |

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>One mark for working, (all three columns P, Q and R)<br>One mark for each correct column Y, Z<br><br>
<table>
<tr><th>A</th><th>B</th><th>C</th><th>P</th><th>Q</th><th>R</th><th>Y</th><th>Z</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr>
</table></td><td>3</td></tr>
<tr><td>7(b)</td><td>Full adder</td><td>1</td></tr>
<tr><td>7(c)</td><td>One mark for each point<br><br>$Y = \overline{A}\ \overline{B}\ C + \overline{A}\ B\ \overline{C} + A\ \overline{B}\ \overline{C} + A\ B\ C$<br>Purpose: Sum bit<br><br>$Z = \overline{A}\ B\ C + A\ \overline{B}\ C + A\ B\ \overline{C} + A\ B\ C$<br>Purpose: Carry output</td><td>4</td></tr>
</table>

| Question | Answer | Marks |
|---|---|---|
| 8(a) | **One** mark for each correct marking point **(Max 2)**<br><br>• The initial order of the data<br>• The number of data items to be sorted<br>• The efficiency of the sorting algorithm | **2** |

| Question | Answer | Marks |
|---|---|---|
| 8(b) | **One** mark for each marking point **(max 6)**<br><br>MP1 Use of `FOR` loop to cycle through the whole year group<br>MP2 Temporary storage of the score being 'inserted'<br>MP3 Temporary storage of the corresponding name elements<br>MP4 Use of `WHILE` loop with correct exit clause<br>MP5 Moving of all three elements of data to next array elements<br>MP6 Correct updating of counter variable<br>MP7 Final insertion of all three data elements<br><br>**Example algorithm**<br>`YearSize ← 249`<br>`FOR Student ← 2 to YearSize`<br>`  Temp1 ← Score[Student]`<br>`  Temp2 ← Name[Student,1]`<br>`  Temp3 ← Name[Student,2]`<br>`  Counter ← Student`<br>`  WHILE Counter > 1 AND Score[Counter - 1] < Temp1`<br>`  Score[Counter] ← Score[Counter - 1]`<br>`  Name[Counter,1] ← Name[Counter - 1,1]`<br>`  Name[Counter,2] ← Name[Counter - 1,2]`<br>`  Counter ← Counter - 1`<br>`  ENDWHILE`<br>`  Score[Counter] ← Temp1`<br>`  Name[Counter,1] ← Temp2`<br>`  Name[Counter,2] ← Temp3`<br>`NEXT Student` | **6** |

| Question | Answer | Marks |
|---|---|---|
| 9(a) | **One** mark for each correct marking point **(Max 2)**<br><br>• Imperative languages use variables<br>• … which are changed using (assignment) statements<br>• … they rely on a method of repetition / iteration.<br>• The statements provide a sequence of commands for the computer to perform<br>• … in the order written / given<br>• … each line of code changes something in the program run. | **2** |
| 9(b) | **One** mark for each correct marking point **(Max 2)**<br><br>• Instructs a program on what needs to be done instead of how to do it<br>• ... using facts and rules<br>• … using queries to satisfy goals.<br>• Can be logical or functional<br>• Logical - states a program as a set of logical relations<br>• Functional – constructed by applying functions to arguments / uses a mathematical style | **2** |

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>9(c)</td><td>One mark for each correct programming paradigm (Max 4)
<table>
<tr><th>Program code example</th><th>Programming paradigm</th></tr>
<tr><td><pre>male(john).
female(ethel).
parent(john, ethel).</pre></td><td>Declarative</td></tr>
<tr><td><pre>FOR Counter = 1 TO 20
    X = X * Counter
NEXT Counter</pre></td><td>Procedural / imperative</td></tr>
<tr><td><pre>Start: LDD Counter
       INC ACC
       STO Counter</pre></td><td>Low-level / assembly</td></tr>
<tr><td><pre>public class Vehicle
{
    private speed;
    public Vehicle()
    {
        speed = 0;
    }
}</pre></td><td>Object oriented / (OOP)</td></tr>
</table></td><td>4</td></tr>
</table>