OCR
Oxford Cambridge and RSA

GCSE (9–1)

Computer Science

J276/02: Computational thinking, algorithms and programming

General Certificate of Secondary Education

Mark Scheme for November 2020

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 2020

Annotations

Annotation	Meaning
SEEN	Answer seen – this annotation **must** be used on all blank pages within an answer booklet (structured or unstructured) and on each page of an additional object where there is no candidate response.
	Omission mark
	Benefit of doubt
	Cross
	Follow through
	Not answered question
	Benefit of doubt not given
	Repeat
	Tick

Each question must include annotation. All marks credited on point-based marking must be given a tick.

<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 for each letter

<table><tr><td>Decomposition</td><td>D</td></tr><tr><td>Abstraction</td><td>B</td></tr><tr><td>Input Sanitisation</td><td>A</td></tr><tr><td>Casting</td><td>F</td></tr></table></td><td>4

AO1
1a(4)</td><td>Accept answers that write the definition instead of the letter.</td></tr>
<tr><td>1</td><td>(b)</td><td>(i)</td><td>• <code>timer = 7.3</code></td><td>1

AO3
2b(1)</td><td>Ignore dim / define / etc and data types
Do not allow use of string delimiters or other unsuitable data types.
Allow other suitable assignment symbols (e.g. :=) Do not allow == for assignment.
Do not penalise case. Spelling must be accurate</td></tr>
<tr><td>1</td><td>(b)</td><td>(ii)</td><td>• Real // Float</td><td>1

AO2
1b(1)</td><td>Allow decimal, single, double or equivalent</td></tr>
<tr><td>2</td><td>(a)</td><td></td><td><table><tr><th>Line</th><th>Program code</th><th>Output</th></tr><tr><td>08</td><td><code>print score</code></td><td>18</td></tr><tr><td>09</td><td><code>print "name"</code></td><td>name</td></tr><tr><td>10</td><td><code>print newscore(score,2)</code></td><td>37</td></tr><tr><td>11</td><td><code>print score</code></td><td>18</td></tr></table></td><td>4

AO2
1b(4)</td><td></td></tr>
</table>

2	**(b)**		1 mark per bullet to max 2 • **Easier/quicker** for humans to **write** • **Easier/quicker** to **read / understand / remember** • **Easier/quicker** to **maintain / debug / spot errors** • …because code is closer to English / uses English words • Less code to write • …because one HLL instruction represents many assembly instructions • Portable (between processors) // will work with different types of computer	2 AO1 1b(2)	Accept "human language" as English for BP4 "Easier to use" is too vague.
2	**(c)**		1 mark per bullet to max 2 • Each character (in character set) has a **unique** (binary) number/value • Each character in the **string** is assigned its associated number/value • The (binary) value of each character is stored/combined (in order) • … by example e.g. The binary value for D, then for r, then for u • Uses ASCII/Extended ASCII/Unicode	2 AO2 1a(2)	

<table>
<tr><th colspan="3">Question</th><th>Answer</th><th>Mark</th><th>Guidance</th></tr>
<tr><td>3</td><td>(a)</td><td>(i)</td><td>1 mark per bullet to max 2
e.g.
• Check the program meets the user requirements
• Check the program works (as intended) // detect logic / syntax errors
• Check the program does not crash (under invalid entry) // check error messages are suitable
• …allow these errors to be fixed
• …make sure there are no problems when released
• Any suitable example related to the vending machine e.g. gives correct change</td><td>2

AO1
1b(2)</td><td>Allow two any suitable examples for two marks

BOD “find errors”, “find bugs” for BP2

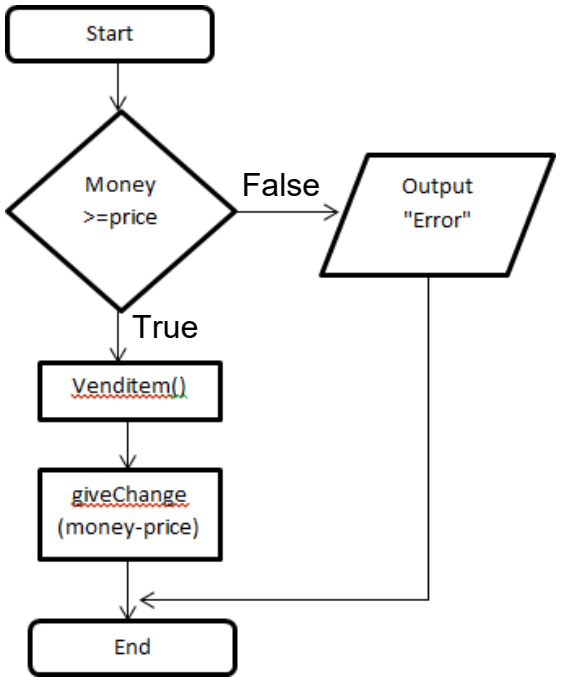
“fix errors” by itself is one mark (BP4)</td></tr>
<tr><td>3</td><td>(a)</td><td>(ii)</td><td>1 mark per bullet to max 2
• Iterative is during development // repeatedly testing after/while making changes
• Final is when the development is (almost) complete // done after iterative testing</td><td>2

AO1
1b(2)</td><td>Do not accept just “repeatedly testing” for iterative

BOD “iterative testing tests modules/sections”</td></tr>
<tr><td>3</td><td>(a)</td><td>(iii)</td><td><table><tr><th>Code entered</th><th>Money inserted</th><th>Expected result</th></tr><tr><td></td><td></td><td></td></tr><tr><td>C2</td><td></td><td></td></tr><tr><td></td><td>£0.49
(or any value less that £0.50)</td><td></td></tr><tr><td></td><td></td><td>Invalid Selection
(or any suitable error message)</td></tr></table></td><td>3

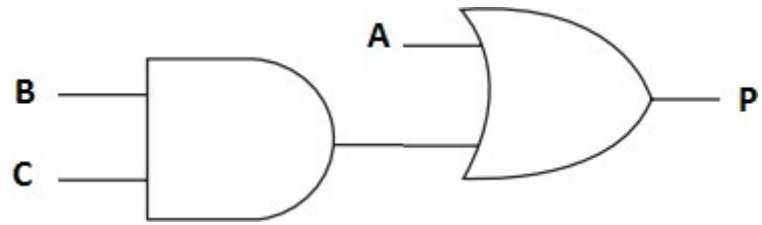
AO3
2b(3)</td><td>For £0.49 accept any value <£0.50. Must be a specific value, not a description.

Accept any suitable error message for invalid selection</td></tr>
</table>

3	**(b)**	**(i)**	• `money` • `price`	1 AO1 1b(1)	Must be an identifier, not description. Ignore case.
3	**(b)**	**(ii)**	• one	1 AO2 1b(1)	
3	**(c)**		1 mark per bullet • Checking if `money>=price`… • …decision (diamond shape) used • …`venditem()` and `giveChange(money-price)` <u>if True/Yes</u> • …output an error if <u>False / No</u> • Terminator used to start and end the program and all paths terminated	5 AO3 2b(5)	Reasonable attempt at BP1 needed for credit BP2, 3 and 4 Ignore other additional code. BP3 and BP4 must follow on from True/False // Yes/No decision to be credited. Subroutines names and parameters must be correct. Ignore missing brackets on `venditem`. 

Question			Answer	Mark	Guidance
3	d	i	1 mark per bullet to max 2 • Indentation // whitespace • Appropriately named variables / identifiers • Modularisation / use of subroutines	2 AO2 1b(2)	
3	d	ii	• Comments • Use of constants	1 AO2 1b(1)	
3	e		• `SELECT ItemCode` // `*` • `FROM ITEMS` • `WHERE` • …`Stock < 10`	4 AO3 2b(4)	Accept other fields shown in addition to `ItemCode` Accept `Stock <=9` / etc. Ignore case. Spelling of fields and table must be correct. If `WHERE` missing, `Stock < 10` must be after `FROM` clause.

3	f		1 mark per bullet
			• Input from user
			• Check IF input value is "**on**"…
			• … if so, assign **1** to `statevalue`
			• Correct assignment of **2 for "off"** and **3 for "suspended"** with correct state and `IF`
			• Correct logical check (else) to output "invalid state" <u>if no state set</u>
				5
				
				AO3 2b(5)
					Accept alternative error messages. Variable names must not include obvious spaces.
					
					BP3 dependent on BP2. BP2 and BP4 must be a logical comparison using `IF` and not just the CASE statement. NE to simply replace `CASE` with `IF`.
					
					Penalise each error once then apply FT.
					
					e.g.
					`newstate = input("Enter the new state : ")`
					`if newstate == "on" then`
					`  statevalue = 1`
					`elseif newstate = "off" then`
					`  statevalue = 2`
					`elseif newstate = "suspended"`
					`  statevalue = 3`
					`else`
					`  print("Invalid state")`
					`endif`

4	**(a)**		• E 3	2 AO1 1b(2)	1 mark per digit (mark right to left) Max 1 if any additional leading values
4	**(b)**		• 0110 1001 <u>must be 8 bits</u>	2 AO1 1b(2)	1 mark per nibble (mark right to left). Max 1 if any additional leading values
4	**(c)**		1 mark per bullet to max 2 • Easier/quicker to communicate / enter / write / read / remember • Less chance of input errors // easier to spot errors • They are smaller / shorter • Easy to convert between binary and Hexadecimal	2 AO1 1b(2)	Mark response as a whole. Do not accept answers simply describing what hexadecimal is. “easier to understand” or “easier to use” on its own is NE BP3 (smaller) must refer to size when written down, NOT size when stored which is unaffected)
4	**(d)**	**(i)**	• 3	1 AO1 1b(1)	CAO
4	**(d)**	**(ii)**	• 1	1 AO1 1b(1)	CAO
4	**e**		• 00001111	1 AO1 1b(1)	Ignore missing or additional leading zeros
4	**f**	i	1 mark per bullet point • B AND C • OR gate with two inputs, one of which is A • …correct connection of these two gates with no additional gates / connections	3 AO1 1b(3)	Shape must be accurate 

<table>
<tr>
<td>4</td>
<td>f</td>
<td>ii</td>
<td>1 mark per bullet point
• Correct completion of A and B inputs as 1 1
• 0 output for 01 input
• 0 output for 10 input
• 0 output for 11 input</td>
<td>4

AO1
1b(1)
AO2
1b(3)</td>
<td>CAO
<table>
<tr><th>A</th><th>B</th><th>P</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td>
</tr>
</table>

Question			Answer	Mark	Guidance
5	a		• Number of pixels (in an image) • Height and width (of an image)	1 AO2 1b(1)	Accept pixels per inch / mm / unit area (density)
5	b		• 90 (pixels in an image) // 15 x 6 (pixels in image) • Multiply pixels x bits per pixel • …2 bits required per pixel (because 3 colours) • **180** bits overall answer	4 AO1 1b(2) AO1 1b(2)	Must clearly show multiplication for 3rd BP
5	c		• Reduce number of pixels / resolution • Reduce number of colours • Use lossy compression • Use lossless compression	2 AO2 1a(2)	Accept descriptive answers linked to given logo (e.g "change to black and white only") “Make image smaller” is NE Allow compression by itself for one answer.
5	d	i	• Data about data / the image/file // properties of the file	1 AO1 1b(2)	Do not accept examples without a definition.
5	d	ii	e.g. • height • width • colour depth • resolution • geolocation • date/time **created/last edited //** timestamp • file type • author details	1 AO1 1a(2)	Accept any sensible data that could be stored alongside an image. Do not accept filename
6	(a)		• Access “Rob” / studentnames[0]… • …does **not** equal “Anna” // not desired item // move on • Access “Anna” / studentnames[1] • …does equal “Anna” // stop // item found	4 AO2 1b(4)	Answer must refer to this array, not a generic description of linear search. “Access first item” is NE for BP1 or BP3. Must refer to this scenario. Max 1 for “Compare ‘**Anna**’ to **each item** in list” if nothing else credited.

<table>
<tr>
<td>6</td>
<td>(b)</td>
<td></td>
<td>
• Anna inserted before Rob as first two elements…
• …Huw correctly inserted into sorted list…
• …Emma correctly inserted into sorted list …
• …Patrice correctly inserted into sorted list …
• …Iqbal correctly inserted into sorted list and no further changes made.
</td>
<td>5

AO2
1b(5)</td>
<td>
<table>
<tr><td>Rob</td><td>Anna</td><td>Huw</td><td>Emma</td><td>Patrice</td><td>Iqbal</td></tr>
<tr><td>Anna</td><td>Rob</td><td>Huw</td><td>Emma</td><td>Patrice</td><td>Iqbal</td></tr>
<tr><td>Anna</td><td>Huw</td><td>Rob</td><td>Emma</td><td>Patrice</td><td>Iqbal</td></tr>
<tr><td>Anna</td><td>Emma</td><td>Huw</td><td>Rob</td><td>Patrice</td><td>Iqbal</td></tr>
<tr><td>Anna</td><td>Emma</td><td>Huw</td><td>Patrice</td><td>Rob</td><td>Iqbal</td></tr>
<tr><td>Anna</td><td>Emma</td><td>Huw</td><td>Iqbal</td><td>Patrice</td><td>Rob</td></tr>
</table>
Sorted list highlighted
</td>
</tr>
<tr>
<td>6</td>
<td>(c)</td>
<td></td>
<td>
• Use of iteration (any use) …
• ...loops for each item in array // loops 6 times
• …to print out each item in <code>studentnames</code>
• …input attendance
• Add up/calculate students present and absent
• …Outputs present and absent (in suitable message)
</td>
<td>6

AO3
2b(6)</td>
<td>
BP 2 and 3 may be met together with suitable input statement. Both dependent on attempt at iteration.

BP5 not dependent on correct previous parts.

BP6 needs reasonable attempt at totalling present and absent figures.

Ignore non-initialisation of counter variables.

Flowcharts are acceptable but must show how to solve the problem, not simply repeat the question.

<u>Example algorithm</u>

<pre>
present=0
absent=0
for i = 0 to (studentnames.length) -1
    print(studentnames[i])
    attendance=input("absent or present?")
    if attendance=="present" then
        present=present+1
    else
        absent=absent+1
    endif
next i
print ("Present students: " + present)
print ("Absent students: " + absent)
</pre>
</td>
</tr>
</table>

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

Cambridge Assessment