Chhetri Academy

chhetriacademy.com

GCSE COMPUTER SCIENCE

2.3

OCR J277 | Specification point 2.3.2

Testing

Untiered

Answers and examiner comments

3	2017-2025	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.

Original question 3

The full source mark scheme begins on page 19. Use the original paper code, tier and question number shown above.

Question			Answer	Mark	Guidance
5	(a)	(i)	1 mark each to max 2 • 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	**2** (AO1 1b)	Allow answers that explain what would happen if not tested (e.g. “there might be bugs”)

Question			Answer	Mark	Guidance
5	(a)	(ii)	1 mark for name, 1 mark for matching description e.g. • 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… • …test using data that should be **rejected** / is not acceptable / causes an error	**2** (1 AO1 1a), (1 AO2 1b)	Allow other sensible descriptive names for testing. Description must match test type. Must be a description and not just an example, but example may support description. Do not accept descriptions that simply repeat type of test without further clarification (e.g. “boundary, testing the boundary”). Allow : • Black box testing… • …testing **without** access / knowledge of a system’s workings. • White box testing… • …testing **with** access / knowledge of system’s workings. Allow other sensible / valid types of testing. Do not accept examples of validation (e.g. type test, range check) “data that is not expected” is NE for invalid/erroneous unless clarified.

Question			Answer	Mark	Guidance
5	(a)	(iii)	1 mark for feature 1 mark for matching description e.g. • 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…	**4** (AO2 1b)	Allow other sensible names for features. 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. If compiler and interpreter given as two distinct features, allow both (with suitable descriptions). Do not allow translator and compiler/interpreter. Description must match feature. “finding errors” is NE for description of error reporting. Allow sensible references to AI where appropriate. Sensible description of use needed. Allow other sensible features of an IDE (e.g. line numbering, auto indent, collapsed blocks, etc) with suitable description. For text editor / error diagnostics / debugger, allow other sensible features listed as features in the mark scheme as description (e.g. “text editor,

Question			Answer	Mark	Guidance
			• … 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.		provides pretty printing", "debugger, provides stepping")

Question			Answer	Mark	Guidance
5	(b)		1 mark for method, 1 mark to max 2 for each use e.g. • 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 • … **limit** number of **characters** // check **maximum** / **minimum string** length • … **answer / input** must be **1 or 2 characters**	**6** (4 AO2 1b) , (2 AO1 1a)	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). Do not accept uses that simply repeat the name of the check (e.g. “range check, checks a range of numbers”) 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). 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). Accept alternative names or descriptions (e.g. existence check, boundary check) but name of check must be sensible. Mark each answer as a whole, ignore method/use headings. Do not accept defensive design elements (e.g. input sanitisation, authentication)

Question			Answer	Mark	Guidance
			• …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		Examples of program code can be actual code (e.g. `if inp>=2 and inp<=20`) 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.

Question			Answer	Mark	Guidance

5	(c)		1 mark each to max 6
			
			• Initialise / declare `score` (to zero) before use, outside of any loop
			• Generates 2 random numbers between 1 and 10
			• Inputs answer from user displaying suitable numbers
			• Checks if input is correct answer…
			• … if correct adds 1 to `score`
			• Repeats BP2 to 5 three times (for bullet points attempted)
			• Outputs `score` after reasonable attempt at counting
				6
				(AO3 2b)
					No need to cast data to string/integer.
					
					If random numbers chosen, BP3 must use these. If no random numbers chosen, allow manually setting values
					
					BP6 can be awarded for either a loop repeating 3 times or the same code written out 3 times
					
					BP5 can be given FT if sensible attempt at BP4
					
					Do not award BP6 if same numbers used for every question. Must pick new values each time.
					
					Do not penalise potential off by 1 errors for looping (Python) or random number generation
					
					Example answer
					
					`score = 0`
					`for count = 1 to 3`
					`  num1 = random(1, 10)`
					`  num2 = random(1, 10)`
					`  ans = input("What is" +num1 +  " + " + num2 + "?")`
					`  if ans = num1 + num2 then`
					`  score = score + 1`
					`  end if`
					`next count`
					`print("You scored " + score)`

Section B

<table>
<tr>
<td>6</td>
<td>(a)</td>
<td></td>
<td>1 mark for each row

<table>
<tr><th>Variable</th><th>Boolean</th><th>Char</th><th>String</th><th>Integer</th><th>Real</th></tr>
<tr><td>UserName</td><td></td><td></td><td>✓</td><td></td><td></td></tr>
<tr><td>EmergencyPhone</td><td></td><td></td><td>✓</td><td></td><td></td></tr>
<tr><td>DoorSensor</td><td>✓</td><td></td><td></td><td></td><td></td></tr>
<tr><td>DoorTime</td><td></td><td></td><td></td><td>✓</td><td></td></tr>
</table>
</td>
<td>4
(AO3 2a)</td>
<td>No mark if more than 1 tick on a row.

Allow other indications of choice (e.g. cross) as long as clear.</td>
</tr>
<tr>
<td>6</td>
<td>(b)</td>
<td></td>
<td>1 mark each:
• Attempt at using selection / condition controlled loop
• Checking if system armed // while system armed
• If Door Sensor active OR Window Sensor active (both checks required)
• calling SoundAlarm correctly</td>
<td>4
(AO3 2b)</td>
<td>Selection could be done using IF statement, case statement or any other sensible valid method.

Allow reference to <code>AlarmActivated</code> or equivalent instead of <code>SystemArmed</code>

Ignore any inputs or modification of variables.

Allow True / False as strings. Allow checking against strings (e.g. <code>if SystemArmed == "active"</code>)

Allow checking armed/disarmed for BP2 and BP3

Only award BP4 if SoundAlarm correctly called / not called in every situation. If issues on previous lines (e.g. lack of brackets where needed) means this is not the case, do not award BP4.</td>
</tr>
</table>

					Checking could be done by evaluating variable directly (`if SystemArmed`) or by comparison (`if SystemArmed == True`)
					
					<u>Example answer 1</u>
					
					`if SystemArmed then`
					`  if DoorSensorActive then`
					`  SoundAlarm()`
					`  else if WindowSensorActive then`
					`  SoundAlarm()`
					`  endif`
					`endif`
					
					<u>Example answer 2</u>
					
					`while SystemArmed then`
					`  if DoorSensorActive then`
					`  SoundAlarm()`
					`  else if WindowSensorActive then`
					`  SoundAlarm()`
					`  endif`
					`endif`
					
					<u>Example answer 3</u>
					
					`if SystemArmed and (DoorSensorArmed or WindowSensor) then`

					`SoundAlarm()`
					`endif`
					
					Note – above example needs brackets,
					
					`if SystemArmed and DoorSensorArmed or WindowSensor then`
					
					is not logically valid for this scenario (will sound alarm when not armed if window sensor is active)
					
					<u>Example answer 4</u>
					
					`if SystemArmed and DoorSensorArmed`
					`SoundAlarm()`
					`else if SystemArmed and WindowSensorArmed`
					`SoundAlarm()`
					`endif`

6	**(c)**	**(i)**	1 mark for • Line 04	**1** (AO3 1)	
6	**(c)**	**(ii)**	1 mark from • `sensorType` • `sensorNumber` • `sensorID`	**1** (AO3 1)	Do not penalise case, spacing or minor misspellings.
6	**(c)**	**(iii)**	1 mark for • Boolean	**1** (AO3 1)	Ignore minor misspelling. Accept Bool.

6	**(c)**	**(iv)**	1 mark from • Line 01 • Line 02 • Line 03 • Line 05	**1** (AO3 1)	
6	**(c)**	**(v)**	1 mark each • Selection • Sequence	**2** (AO3 1)	Ignore minor spelling errors / differences Do not accept examples (e.g. IF)
6	**(d)**		1 mark each • `SELECT SensorID // SELECT *` • `FROM events` • `WHERE Length > 20 AND sensorType = "Door"` // `WHERE sensorType = "Door" AND Length > 20`	**3** (AO3 2c)	Max 2 if out of order or anything extra that affects the output. BP1 can select multiple fields as long as `SensorID` is included. Ignore case. Only penalise spaces if obvious. Field names must be correct. "door" must be in quotation marks for BP3. Allow quotation marks for field names and table name BP3 can use == or = for equivalence. Allow alternative WHERE clauses that are logically correct (e.g. `WHERE length >=21`)

6	**(e)**		1 mark each
			• **Define** procedure SaveLogs…
			• …with **two** valid parameters
			• Open file (for write/append) …
			• … using the file name **passed in as parameter**
			• Write data **to file**…
			• …using the data **passed in as parameter**
			• Close file
				6
				(AO3 2b)
					Must be clear that answer is a procedure definition, do not credit calling procedure for BP1. Allow function definition.
					
					If parameters are later overwritten, do not credit BP2 but FT for BP4 and 6.
					
					Closing text file does not need reference to file name/object – e.g. "`close file`" is enough. However, if given reference must be correct.
					
					If code given outside of procedure, do not give BP4 and BP6
					
					Allow FT for multiple occurrences of same mistake (e.g. not using filename correctly for open and close)
					
					<u>Example answer</u>
					
					`procedure SaveLogs(data, filename)`
					`  logFile = open(filename)`
					`  logFile.writeLine(data)`
					`  logFile.close()`
					`endprocedure`

6	**(f)**	**(i)**	1 mark for: • Casting / cast	**1** (AO3 2a)	Accept type casting Do not accept conversion. Do not accept examples of casting.

6	**(f)**	**(ii)**	1 mark each to max 6 • Input date and store in variable / use directly • Access **all** seven (indexes 0 to 6) events in array // loop for each event in array • Attempt at selection… • …to compare `date` input against `date` in array (element 0) • …adding `length` (element 3) from array to the total if dates match. • Outputting calculated total and date in appropriate message(s) at the end	**6** (AO3 2b)	BP2 can be achieved either by iteration accessing each event or manually repeating code to access each event. Must be 0 to 6, not 1 to 7. Allow reference `to events` (table given) or `arrayEvents` (2D array) in answer as long as used consistently. BP2 loop allow off by one errors (Python), looping to array length or array length – 1. Allow `for each item` in array or any other suitable loop. BP4 and BP5 allow array reference as either column major or row major. Output can either be once at the end or on every iteration, as long as it is output at the end. Only give output mark if attempt made to **calculate** total within the algorithm. Do not penalise capitalisation or minor misspellings of variable names.

					Example answer 1
					
					`total = 0`
					`date = input("Please enter date")`
					`for count = 0 to events.length-1`
					`  if events[0, count] == date then`
					`  total = total + events[3,count]`
					`  endif`
					`next count`
					`print("There were " + total + "`
					`events on " + date)`
					
					Example answer 2
					
					`total = 0`
					`date = input("Please enter date")`
					`for item in events:`
					`  if item[0] == date then`
					`  total = total + item[3]`
					`  endif`
					`next count`
					`print("There were " + total + "`
					`events on " + date)`

Need to get in touch?

If you ever have any questions about OCR qualifications or services (including administration, logistics and teaching) please feel free to get in touch with our customer support centre.

Call us on

01223 553998

Alternatively, you can email us on

support@ocr.org.uk

For more information visit

- **ocr.org.uk/qualifications/resource-finder**
- **ocr.org.uk**
- **Twitter/ocrexams**
- **/ocrexams**
- **/company/ocr**
- **/ocrexams**

CAMBRIDGE
UNIVERSITY PRESS & ASSESSMENT

OCR is part of Cambridge University Press & Assessment, a department of the University of Cambridge.

For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored. © OCR 2023 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 operates academic and vocational qualifications regulated by Ofqual, Qualifications Wales and CCEA as listed in their qualifications registers including A Levels, GCSEs, Cambridge Technicals and Cambridge Nationals.

OCR provides resources to help you deliver our qualifications. These resources do not represent any particular teaching method we expect you to use. We update our resources regularly and aim to make sure content is accurate but please check the OCR website so that you have the most up-to-date version. OCR cannot be held responsible for any errors or omissions in these resources.

Though we make every effort to check our resources, there may be contradictions between published support and the specification, so it is important that you always use information in the latest specification. We indicate any specification changes within the document itself, change the version number and provide a summary of the changes. If you do notice a discrepancy between the specification and a resource, please contact us.

Whether you already offer OCR qualifications, are new to OCR or are thinking about switching, you can request more information using our Expression of Interest form.

Please get in touch if you want to discuss the accessibility of resources we offer to support you in delivering our qualifications.

Question			Answer	Mark	Guidance
1	(a)		• Normal • Invalid / erroneous • Invalid / erroneous • Boundary // boundary **and** normal	4	Accept either boundary by itself for last row **or** <u>both</u> boundary and normal.
1	(b)	(i)	• `>= 1 // > 0` • **`num`** `<=100 //` **`num`** `< 101` • `else`	3	Allow check either way around (e.g. could check higher boundary first) for BP1 and BP2. BP2 must include reference to **`num`** variable. BP3 Do not allow `elseif / elif` unless full correct logical comparison included (e.g `elseif num <1 or num > 100`) Section A so allow “less than or equal to” etc.
1	(b)	(ii)	• `num`	1	Do not allow any other code around variable identifier. Ignore capitalisation. Ignore minor spelling errors.
1	(b)	(iii)	• `AND`	1	Do not allow any other code around operator. Ignore capitalisation. Ignore minor spelling errors.

OCR
Oxford Cambridge and RSA

GCSE

Computing

Unit **A451**: Computer systems and programming

General Certificate of Secondary Education

Mark Scheme for June 2017

Oxford Cambridge and RSA Examinations

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 will not enter into any discussion or correspondence in connection with this mark scheme.

© OCR 2017

A451 **Mark Scheme** **June 2017**

Annotations

Annotation	Meaning
	Blank Page – 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
	Subordinate clause/Consequential error
	Cross
	Expansion of a point
	Follow through
	Not answered question
	Benefit of doubt not given
	Point being made
	Repeat
	Slash
	Tick
	Too vague
	Zero (big)

A451 **Mark Scheme** **June 2017**

MARK SCHEME

Question			Answer / Indicative Content	Mark	Guidance
1	a		1 mark per bullet to max 2 • Hardware • Software • Allows for input, process and output	2	All of input, process and output required for third bullet. Do not accept examples of hardware/software.
1	b	i	Input: to allow instructions/data/commands/destination to be input/entered/given/example (e.g. entering a post code)	1	Do not accept "information".
1	b	ii	1 mark for input device, 1 for sensible use of device e.g. • Microphone… • …to let the user say their destination/voice recognition • Touch screen… • …let the user press options/enter data • Keyboard / braille keyboard… • …to let the user enter their destination • Camera… • …to let the user enter/select options by moving their eye/use an eye typer • Puff/suck switch… • …to allow the users to select options	2	Allow any reasonable **input** device, do not accept software e.g. voice recognition. Do not accept "sensor" without detailing what type of sensor. Accept "braille keyboard" but do not accept "braille" on its own. Accept buttons. Second mark is awarded for a sensible use of the device – for example, what is inputted (eg the destination) or how it is inputted (eg inputted using your voice). If input device is incorrect, do not award follow on mark for use.
1	c	i	1 mark for why, 1 for application • Long term/permanent/non-volatile/persistent storage • Store maps//user preferences//journey history//operating system//sat nav software	2	Second bullet point must be related to the sat nav, not just generic "programs" or "data".

A451 **Mark Scheme** **June 2017**

1	**c**	**ii**	1 mark for identifying the reason why solid state was chosen, 1 for explanation e.g. • Is durable/robust//no moving parts • Less likely to be damaged//by example (eg dropped) • Faster <u>read/write speed</u> • Faster to boot//faster to load maps//more responsive • Small in size/light in weight • Reduces size of the sat nav// fit inside the sat nav • Lower power requirements • More suitable for running off batteries	**2**	Do not accept portability / reliability. Do not accept "faster" without clarification of what is faster.
1	**d**	**i**	1 mark per bullet to max 1 • <u>Fetch data/instructions</u> • <u>Decode data/instructions</u> • <u>Execute instructions</u> • Performs fetch-(decode)-execute cycle • Performs calculations/arithmetic/logical operations	**1**	Must include fetch / decode / execute as verb for first three bullet points. Must include reference to either data or instructions. Allow reference to tasks carried out by the Control Unit.
1	**d**	**ii**	1 mark per bullet to max 2 • Computers are made of logic gates/transistors/switches… • …which can only be two states / on or off / 1 or 0 /high or low	**2**	Second point dependent on first bullet point being awarded.

A451 **Mark Scheme** **June 2017**

<table>
<tr><td>2</td><td>a</td><td>i</td><td>1 mark
e.g.
• Tags
• Hyperlinks
• Location/dimensions/name of images/videos/sound
• Structure / e.g. table
• Layout instructions
• Formatting e.g font/colour/style
• Metadata</td><td>1</td><td>Accept any valid item that could be in a HTML file e.g. a link to a CSS file

Accept other languages/scripting that could embedded into a HTML document, eg CSS/Javascript/etc.

Do not accept: image, video etc. The HTML file holds a link to the files, not the actual image or video itself.

Mark first answer only.</td></tr>
<tr><td>2</td><td>a</td><td>ii</td><td>1 mark per bullet to max 2
• All webpages are made using HTML
• All browsers read/understand HTML
• Allows for compatibility
• Allows devices to understand the same code/display the webpage correctly</td><td>2</td><td></td></tr>
<tr><td>2</td><td>b</td><td>i</td><td>1 mark for getting 1 correct
2 marks for getting both correct
<table><tr><td></td><td>Red</td><td>Green</td><td>Blue</td></tr><tr><td>Decimal</td><td>111</td><td>58</td><td>156</td></tr><tr><td>Hexadecimal</td><td>6F</td><td>3A</td><td>9C</td></tr></table></td><td>2</td><td></td></tr>
<tr><td>2</td><td>b</td><td>ii</td><td>1 mark per bullet to max 1
• Easier to remember/to enter/to read
• Faster to enter/read
• Fewer digits to remember/to enter/to read
• Less prone to error when entering/reading/communicating
• Quick to convert into binary</td><td>1</td><td>Do not accept “easier to understand”

“Less prone to error” requires further detail as shown.</td></tr>
</table>

A451			Mark Scheme		June 2017
2	**c**		1 mark per bullet to max 2 • Compression makes the file size smaller//takes up less space (on Shannon's computer) • File(s) upload more quickly / lower bandwidth required • File(s) take up less storage space on the web server • File(s) can be downloaded/streamed more quickly by the end user//website will load more quickly	**2**	
2	**d**		1 mark per bullet, to max 3 • URL/domain name is sent to the DNS • DNS has a list/database/table of URLs **and** matching IP addresses. • DNS looks up/translates/resolves/searches/finds the IP address • DNS sends back IP address (to web browser) • If not found, refers to higher/other DNS/returns error • DNS updates from other DNS • Browser uses IP address to retrieve webpage / connect to web server.	**3**	Do not accept "each URL has an associated IP address" on its own.

A451 **Mark Scheme** **June 2017**

3	**a**		1 mark for each pseudocode statement
			
			`Total = Total + NumberArray(`**`Count`**`)`
			
			`Mean =` **`Total/Quantity`**
			`or`
			`Mean =` **`Total/Count`**
			`or`
			`Mean =` **`Total/10`**
				2	Ignore capitalisation.
					
					Accept any correct symbol or structured English meaning division for mean calculation.
					
					Accept mean calculations that refer to 11 numbers:
					e.g.
					• Total/11
					• Total/(Count+1)
					• Total/(Quantity+1)

3	**b**		1 mark per bullet, max 2 for definition, 1 for example Definition: • A location in memory • A value/data that cannot be changed (whilst the program is running) Example: • `Quantity`	**3**	0 marks for "stays the same" / “does not change”. Must have the idea that it cannot / is impossible to change. Correct answer only (“Quantity”) for the example. Do not accept other surrounding code (eg “Const Quantity = 10” is incorrect). Do not accept incorrect spellings. Ignore capitalisation. Do not accept “a constant is a variable that…”
3	**c**		1 mark for data type, 1 for justification Data type: Real/Float/Single/Double/Decimal Justification: can be decimal/fractional/not a whole number	**2**	If candidate uses “decimal” as data type, do not accept “can be decimal” for the justification. Do not award justification if data type is incorrect.
3	**d**	**i**	1 mark per bullet, to max 2 • A construct • Code is executed/run repeatedly//is looped • Until a condition is met/while a condition is true/a set number of times	**2**	Do not accept only an example (eg “for loop”).
3	**d**	**ii**	• While/do while • Repeat/ Repeat until/do until/ Until	**2**	Do not accept “do loop”.

3	**e**		1 mark for sensible borderline data, 1 mark for sensible invalid data. • Borderline – 0, 100 • Invalid – number less than 0 (eg -1, -12) / number more than 100 (eg 101, 206) / non-numeric data (eg “test”, “#!*%”)	**2**	
4	**a**		1 mark per bullet to max 2 • The data is stored permanently / is unchanging / remains when the DBMS or application is closed / non-volatile / on secondary storage • The data has a structure / stored in tables / fields / records	**2**	Accept rows / columns as alternative to records / fields for second bullet point.
4	**b**	**i**	1 mark for data type, 1 for justification Data type: Text/String Reason: leading 0s/will not be treated as a number	**2**	Accept Alphanumeric/Varchar/Char as alternatives for Data type. Do not award reason if data type is incorrect.
4	**b**	**ii**	4	**1**	Correct answer only.
4	**b**	**iii**	1 mark for Field name, 1 for reason Field: RequestID Reason: it will be unique/each request will have a unique number/it will not be repeated/other fields can be repeated	**2**	Do not award reason if field is incorrect. Do not penalise incorrect spelling or capitalisation.
4	**c**	**i**	Max 1 mark per validation rule. Both rules must be different, e.g. cannot both be presence TeacherID e.g. • Presence check//must be entered • Format check//must be letters then numbers • Existence check//must already exist in the database • Lookup//must be selected from a list of valid teachers • Type/character check//must be string • Length check//must be (minimum of) 3 characters long	**2**	Accept any reasonable validation rule for a field Accept name of a rule, or example Mark first answer only if candidates have provided multiple validation techniques for each.

			Date e.g. • Range check//must be within certain dates • Presence check//must be entered • Format/character check//must be DD/MM/YYYY • Type check//must be a valid date • Length check//must be 8/10 characters • Lookup/existence check//must be a valid date (eg from calendar)		
4	**c**	**ii**	3 marks for each feature, 1 for identifying, 1 for description, 1 for example use e.g • Query • Use to select specific information // find records that match a criteria // search for records // extract data • e.g. Find all requests made on a specific date • Form • User friendly way to enter data // uses drop down boxes etc. • e.g. form to enter a new request • Report • User friendly/formatted copy of results // can be used as a hard copy // method of outputting data • e.g. report of all requests made by one teacher for printing • Security • Stop unauthorised access or modification • e.g. usernames and passwords	**6**	The feature identified must be a feature of a DBMS and not a database. Allow any sensible example that relates to this database. If feature is incorrect, do not award marks for description or example. If feature is poorly identified (eg search instead of query), give credit for matching description / example but not for identification. Mark feature/description/example together. Allow table creation / editor / modules / integrity checks / access control / concurrent access / creating links between tables. Do not accept validation (in the question) Do not accept tables / fields/ key fields (all features of a database, not a DBMS).

5*			Points may include: Ethical issues: • Safer driving as no room for human error • Can machines make snap judgements, taking into account all factors – can impact safety • Cars can react to the facts, i.e. avoid accidents • Can record where people go and when they go – how is this data going to be stored, or used? • Impact on pollution (negative and positives) • Impact on employment (taxis / delivery drivers) Legal issues: • Cars will not break traffic laws, e.g. run red lights • Issues with who is legally the driver • Is a drivers' licence needed? • Who is legally liable/responsible in an accident? • Insurance requirements • Data Protection issues with data generated • Possibility of "hacking" into cars • May need to change or adapt existing laws Many points can be considered from both an ethical and legal point of view. Candidates may refer to other valid scenarios of automated technology, not just driverless cars.	**6**	**High Level Response (5-6):** A detailed discussion of at least one ethical and one legal issue, with clear explanations that are linked to the scenario. There will be few if any errors in spelling, grammar and punctuation. Technical terms will be used appropriately and correctly. **Medium Level Response (3-4):** A description of an ethical and/or a legal issue with some explanation/justification that may be weak at times. Material may not be explicitly linked to the context. There may be occasional errors in spelling, grammar and punctuation. Technical terms will be mainly correct. **Low Level Response (1-2):** There is an attempt to describe either a legal issue or ethical issue. The points are poorly expressed or are not related to the context. There is limited, if any, use of technical terms. Errors in grammar, punctuation and spelling may be intrusive. 0 marks, response not worthy of credit

6	**a**		Max 2 per explanation, 1 for example e.g. • Share files between computers/employees // by example • Centralised data storage // by example • Allows for collaborative working // by example • More efficient working practices // by example • e.g. OCR Accounts use a shared drive to share work • Share devices / printers / routers // by example • Fewer peripherals / printers required// by example • Save money in purchasing equipment// by example • e.g. OCR Accounts only needs 1 printer, rather than 1 per employee • Improved communication // by example • Can use e-mail/other appropriate means // by example • e.g. OCR Accounts can e-mail documents/work • Central management of security // by example • Anti-virus/firewall/security is configured for all computers // by example • e.g. each computer at OCR Accounts does not need its own security • Centralised backups // by example • Can backup all computers automatically // by example • e.g. OCR Accounts can backup data for all staff • • Centralised updates // by example • Can update all computers at the same time // by example • e.g. OCR Accounts can update software without having to visit everyworkstation.	**6**	Accept any other reasonable benefit for using a network in this scenario, except for monitoring. Mark question as a whole. 3rd mark is awarded for example linked to the scenario, so max 2 per benefit if no example given.

A451			Mark Scheme		June 2017
6	b	i	Modem/Router	1	

6	**b**	**ii**	max 2 marks per hardware device e.g. • NIC… • …to connect Ethernet cable to computer • Router… • …to receive and transmit data within the network/to send data around a network/to join networks together/to connect to the Internet • Bridge… • …connect networks together • Switch… • …to connect multiple devices together / directs traffic to its destination. • Hub… • …to connect multiple devices together. • Server… • …to store the data/manage the network/store backups • Wireless Access Point/WAP… • …to allow for wireless transmission of data • Cables… • …to connect devices together • Modem… • …to connect computers via telephone lines // to covert digital data to analogue / to convert analogue data to digital.	**4**	Accept any hardware device that can be used to create/set up/produce a network. Device must be different than answer given in 6bi Accept repeater / range extender / powerline adaptor etc.

6	**c***		Points may include:
			• Acceptable use
			  ○ Define what employees can/cannot do on a network
			  ○ Restrict employees to only performing work tasks
			  ○ Stop illegal activities such as accessing inappropriate material, downloading software
			• Disaster recovery
			  ○ A plan in case something happens to the computers/network/data
			  ○ A plan to allow recovery to avoid downtime/restore the network/data
			• Failover
			  ○ Backup devices ready to automatically take over if a device fails
			  ○ Allows the company to continue working
			  ○ Avoids downtime
			• Backup
			  ○ A plan for when data is backed up
			  ○ How it is backed up
			  ○ Who is responsible for it
			  ○ Where it is backed up etc.
			• Archiving
			  ○ Removing old data that is no longer needed but kept in case needed
			  ○ Free up memory space in a computer
			  ○ Plan of when data will be archived, what will be archived etc.
			• Security
			  ○ Levels of access
			  ○ Usernames / passwords / authentication
			  ○ Encryption
				6	**High Level Response (5-6):**
					A detailed discussion of network policies, covering points from at least **two** different policies, with clear explanations of purpose that are linked to the scenario.
					There will be few if any errors in spelling, grammar and punctuation. Technical terms will be used appropriately and correctly.
					
					Medium Level Response (3-4):
					A description of some network policies, covering points from at least **two** policies, with some explanation/justification that may be weak at times. Material may not be explicitly linked to the context. There may be occasional errors in spelling, grammar and punctuation. Technical terms will be mainly correct.
					
					Low Level Response (1-2):
					There is an attempt to describe at least **one** network policy. The points are poorly expressed or are not related to the context. There is limited, if any, use of technical terms. Errors in grammar, punctuation and spelling may be intrusive.
					
					0 marks, response not worthy of credit

7			1 mark per bullet:
			• Storing a number for the user to guess
			• Loops 10 times correctly
			• Inputs the user's guess
			• If correct, outputs congratulations **and** stops the loop / ends the game (any appropriate method of breaking out of loop)
			• If the guess is greater than stored number, outputs lower (or similar)
			• If the guess is lower than stored number, outputs higher (or similar)
			
			e.g. using while loop
			`num = 50 //(could be a random number)`
			`x = 0`
			`while x < 10`
			`  input guess`
			`  if guess == num then`
			`  output "Congratulations"`
			`  x = 10`
			`  elseif guess > num then`
			`  output "lower"`
			`  else`
			`  output "higher"`
			`  endif`
			`x = x + 1`
			`end while`
			
			e.g. example using for loop
			`num = 50 //(could be a random number)`
			`for x = 1 to 10`
			`  input guess`
			`  if guess == num then`
			`  output "Congratulations"`
			`  end //(could be break / exit, or x = 10)`
			`  elseif guess > num then`
			`  output "lower"`
			`  else`
				6	Allow pseudocode, flowchart, or structured English as long as it is not just repeating the instructions and where it meets the bullet points.
					
					If candidate uses FOR loop, accept 0 to 9 / 0 to 10 / 1 to 10 / 1 to 11 (or equivalent) as valid for 2nd bullet point.

			output "higher" endif next		

OCR (Oxford Cambridge and RSA Examinations)
1 Hills Road
Cambridge
CB1 2EU

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; 1 Hills Road, Cambridge, CB1 2EU
Registered Company Number: 3484466
OCR is an exempt Charity

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

© OCR 2017

PART OF THE CAMBRIDGE ASSESSMENT GROUP

LLOYD'S REGISTER QUALITY ASSURANCE ISO9001 UKAS MANAGEMENT SYSTEMS 001

Question 5 (a) (i)

5 Charlie is developing an adding game. The rules of the game are:

- the player is asked 3 addition questions
- each question asks the player to add together two random whole numbers between 1 and 10 inclusive
- if the player gets the correct answer, 1 is added to their score
- at the end of the game their score is displayed.

(a) Charlie has been told that the game will need to be tested before giving it to the players.

(i) Explain why programs should be tested before use.

.....

.....

.....

..... **[2]**

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.

The majority of responses demonstrated an understanding of this and achieved highly.

Question 5 (a) (ii)

(ii) Complete the table by naming and describing **one** type of test that should be used on Charlie's program before releasing it.

Test type	Description

[2]

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.

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.

Assessment for learning



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.

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 they do not appear in this specification.

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).

Question 5 (a) (iii)

(iii) Complete the table by identifying **and** describing **two** features of an IDE that can be used when testing a program.

Feature	Description

[4]

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.

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.

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.

Question 5 (b)

(b) Validating inputs can reduce errors when a program is being run.

Identify **two** methods of validation **and** explain how they can be used on this game.

Validation method 1 .....

Use .....

.....

.....

Validation method 2 .....

Use .....

.....

.....

[6]

Responses which focused on the explicit link to the game described in this question tended to do well.

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.

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.

Misconception

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.

Where candidates suggested validating inputs to allow between 1 and 10, not all marks available were given due to this misconception.

Question 5 (c)

(c) Write an algorithm to play this game. The rules are repeated from the start of the question here:

- the player is asked 3 addition questions
- each question asks the player to add together two random whole numbers between 1 and 10 inclusive
- if the player gets the correct answer, 1 is added to their score
- at the end of the game their score is displayed.

.....

.....

.....

.....

.....

..... **[6]**

As this question appears in Section A, candidates are free to respond in any suitable way, including using flowcharts, structured English, pseudocode or a high-level language.

The majority of high scoring responses used a high-level language consistently.

Where flowcharts or structured English were used, responses needed to clearly show the steps to be taken and not simply repeat the question to achieve marks.

The given question is already decomposed for candidates and many were able to use these bullet points to build a solution that achieved the majority of marks available.

Many responses used random number generation and iteration to create an elegant response that met all mark points. This was pleasing to see and it is extremely encouraging that candidates can use techniques such as these where appropriate without being prompted.

Other responses manually repeated asking the required questions; on this occasion, these were also credited and could have achieved full marks.

Where a mistake was made in one section (such as with iteration), examiners were instructed to use FT (follow through) where possible. This allowed candidates to score marks in later sections if their responses were logically constructed. This is to be fair to candidates so that mistakes are only penalised once in any given question.

A significant number of responses did not access many marks in this question. This would suggest that more practical programming time in lessons would be beneficial.

Section B overview

Section B consists of multiple questions. Questions are based on a given scenario. This year the scenario was a security company.

Questions will always be set within a single scenario for Section B.

Some questions require candidates to respond using either a high-level language that has been studied or OCR Exam Reference Language. If a candidate responded to these questions using flowcharts or structured English, no marks were given.

It is important for centres and candidates to understand that examiners **do not** penalise answers if they do not work in a particular language. Candidates do not state which language they use. Therefore, answers are marked for their logical correctness and consistency.

This includes where answers combine features from multiple high-level languages.

Question 1 (a)

1 A program asks a user to enter a number. Only integer values between 1 and 100 (inclusive) are accepted. The program is tested.

(a) Tick (✓) **one or more** boxes in each row to identify the type or types of test for each example of test data.

Example test data	Normal test data	Boundary test data	Invalid / erroneous test data
27			
"Hello"			
105			
100			

[4]

This question assessed candidates' understanding of different types of test data: normal, boundary, and invalid/erroneous.

The vast majority of candidates performed very well on this question, demonstrating a secure understanding of the definitions and application of each test data category. It was clear that this topic had been well taught across centres, with many candidates achieving full marks.

Use of Invalid/erroneous test data

Although the specification defines invalid data and erroneous data separately, questions in examinations will always group these together as data that a system would reject, as shown in this question.

Question 1 (b) (i)

(b) An algorithm is written for the program.

(i) Complete the algorithm for the program.

```
num = input("Enter a number between 1 and 100")

if num ..... AND ..... then

      print("accepted")

.....

      print("not accepted")

endif
```

[3]

Most candidates were able to correctly identify the purpose of the IF statement and therefore complete the third gap (**`else`**) with ease. Many candidates also correctly completed the first gap as a simple comparison with the variable **`num`** that was already given.

However, the second gap proved more challenging. A common mistake was to write **`<= 100`** without referencing the variable a second time, which is not syntactically valid.

Another frequent error involved misunderstanding the inclusivity of the boundaries. Some candidates used **`num > 1 AND num < 100`**, which would **exclude** valid inputs of 1 and 100. This suggests a need for further emphasis on the difference between inclusive and exclusive comparisons when validating input ranges.

Misconception



Selection statements using two conditions joined together with a Boolean condition such as AND must make sure that both conditions are logically valid comparisons. In many cases, this will mean referencing a variable twice, once in each condition.

For example, **`if num >= 1 AND <= 100`** is incorrect because the second condition is **not** a valid comparison - the value **`num`** must be included in the second part of the comparison.

The correct statement would be **`if num >= 1 AND num <= 100`**.