2.2 Programming Fundamentals – Past Exam Questions – Solutions

2022

<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
<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
(AO2 1b)</td><td>No mark given if both boxes in a row ticked.

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</code> // <code>score +=1</code> // <code>score++</code></td><td>1
(AO3 2b)</td><td>Allow other logically correct answers that result in <code>score</code> increasing by one and being overwritten. Do not accept <code>score + 1 / score = +1</code>

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"

Ignore any superfluous code that does not affect the outcome</td></tr>
<tr><td>2</td><td>(d)</td><td>(i)</td><td>• Convert/change one data type to another
• Line 03 // 3 // three</td><td>2
(AO1 1b, AO2 2b)</td><td>Do not accept "change to string" – this is the use in this example but not a definition.</td></tr>
<tr><td>4</td><td>(b)</td><td>(i)</td><td>• Multiplication
• Division</td><td>2
(AO1 1a)</td><td>Accept other correct answers that mean the same
Accept floor / integer division // division with no remainder (Python v2.x)</td></tr>
<tr><td>5</td><td>(a)</td><td>(i)</td><td>• Integer
• String</td><td>2
(AO3 2a)</td><td>Accept other valid data types from high-level languages (e.g. byte, short for integers)

Do not accept descriptions (e.g. "whole number", "text"). Do not accept "character(s)" for string.</td></tr>
<tr><td>5</td><td>(a)</td><td>(ii)</td><td>• stayComplete</td><td>1
(AO3 2a)</td><td>Ignore spaces or misspelling as long as recognisable.</td></tr>
<tr><td>5</td><td>(a)</td><td>(iii)</td><td>• <code>SELECT FirstName, Surname, Nights, Room, StayComplete</code> // <code>SELECT *</code>
• <code>FROM TblBookings</code>
• <code>WHERE</code>
• <code>Nights > 1</code> // <code>Nights >= 2</code> // <code>Nights BETWEEN 2 AND 5</code></td><td>4
(AO3 1, AO3 2c)</td><td>Order of fields for BP1 not important but must show all fields and be separated by commas.

Ignore capitalisation and spacing. Spelling must be correct. Ignore quotes around numeric values or field/table names.

Allow other logically valid SQL statements. Check with TL if required.

Ignore reference to <code>stayComplete</code> or other valid SQL code that would not affect output.

Max 3 if in wrong order or if includes any extra invalid code</td></tr>
</table>

5	(c)	(i)	• Function header for **newPrice**…
			• …taking (at least) **two** parameters
			• …correctly calculates price based on parameters (if present) within function …
			• …**returns** this calculated price
				4
				(AO3 2b)
					BP1 must be clear that a new **function** is being defined. E.g. `function` / `def` keyword. Allow FT for subsequent marks if not present.
					
					Ignore any code outside attempt at function definition.
					
					Ignore additional parameters. Ignore inputs or additional code as long as these do not overwrite parameters or affect operation of function.
					
					If inputs used instead of parameters, FT for BP3. Allow use of `else` for second room type in BP3.
					
					Attempt at calculation needed to award BP4. Must return (not output) value. Return can be done e.g. in VB by assigning to function name (e.g. `newPrice = price`)
					
					e.g.
					`function newPrice(nights, room)`
					`  if room == "basic" then`
					`  price = 60 * nights`
					`  elseif room == "premium" then`
					`  price = 80 * nights`
					`  endif`
					`  return price`
					`endfunction`

5	(c)	(ii)	• Call function newPrice…
			• …with (`"premium", 5`) as parameters
			• …Output **returned value**
				3
				(AO3 2b)
					Order of parameters not important
					
					"premium" must use string delimiters (e.g. "quotes")
					
					e.g.
					`print(newPrice("premium", 5))`
					
					`x = newPrice(5, "premium")`
					`print(x)`
					
					Do not allow function definitions for BP1
					
					Ignore capitalisation of `newPrice`
					
					Candidate could store returned value in a variable and then print this, or store parameters in variables before passing in – these are all acceptable
					
					Ignore any superfluous code given
					
					Do not credit answers where `newPrice` is overwritten prior to use.
					
					Ignore spaces. Allow function call if brackets missing (e.g. `newprice` instead of `newprice()`)

5	(d)		• For loop changed to **include 0** • total = 0 moved to **before** loop starts / removed	2 **(AO3 2c)**	Allow loop changed to 0 to 8 or 0 to 9 (Python) Do not accept moving total outside loop, NE (could be moved to after loop which would still be a logic error). Do not accept move to top of loop. Accept corrected code shown. Accept reference to count variable limits for BP1.

Sample Paper

3			• SELECT **StudentName, Subject, Grade** • FROM **Results** • WHERE **Subject = "Art"**	**1 (AO1 1b)** **2 (AO3 2a)**	Correct Answer Only Accept SELECT *
7	b		• radius • area	**2 (AO1 1b)**	1 mark per bullet up to a maximum of 2 marks.
	c	i	• 3.142 • 2 • 1 • 30	**1 (AO2 1a)**	1 mark for one correct identification.
	c	ii	• The number does not need to be changed while the program is running • The number can be updated once and it updates throughout	**1 (AO1 1a)**	Maximum of 1 mark.
	d		• HAS been used • HAS been used • HAS NOT been used	**3 AO2 1b**	
8	a		Integer (1)… • …number of seconds not important (1) • … level of accuracy not needed so round to nearest minute (1) • …using a decimal to store seconds (0-60) is not appropriate (1) Real (1)… • … number of seconds may be important (1) • … allows parts/fractions to be stored over integers (1)	**1 (AO3 2a)** **1 (AO3 1)**	One mark for appropriate data type identified. One mark for appropriate justification linked to the data type chosen.
8	c		`print (minsPlayed[0,4])`	**1 (AO3 2b)**	**High-level programming language / OCR Exam Reference Language response required** Do not accept pseudocode / natural English. `print` may be a suitable output command word that could be found in a HLL e.g. `print` (Python), `console.writeline` (VB), `cout` (C++) The array elements may be accessed together `[0,4]` (VB.NET) or separately `[0][4]` (Python)

8	d		• Initialises total as 0 **and** prints out total the end (as per original program)
			• Uses iteration, e.g. FOR, WHILE
			• …that repeats 5 times
			• …correctly adds up values using loop index
			
			e.g.
			`total = 0`
			`for x = 0 to 4`
			`  total = total + hoursplayed[2, x]`
			`next x`
			`console.writeline(total)`
			
			e.g.
			`total = 0`
			`for x in range (0, 4)`
			`  total += hoursplayed[2][x]`
			`next x`
			`print (total)`
				4 (AO3 2c)	**High-level programming language / OCR Exam Reference Language response required**
					
					Do not accept pseudocode / natural English.
					
					MP1 must have appropriate identifier, = and then the numeric 0
					MP2 must have `for` or `while`
					MP3 must have the for stopping condition 4/5
					MP4 must have the same identifier for MP1 and equal and + to add the data in the array (using either `[x,y]` or `[x][y]`. This could be `total = total +` …. Or `total +=` ….

g	ii	• Program calls function correctly using hours and minutes variables
		• Parameters used appropriately
		• Calculation is computed accurately
		• Final total is returned suitably
			4 (AO3 2a)	`hours = input("Please enter number of hours played")`
				`minutes = input("Please enter number of minutes played")`
				`finalTotal = totalMins(hours, minutes)`
				`print (finalTotal)`
				
				`function totalMins(hours,minutes)`
				`  total = (hours * 60) + mins`
				`  return total`
				`endfunction`
				
				1. Parameters named in function must be used within the function itself
				2. Does not matter if function uses different names to those declared in main program
				3. Return must be included with the correct local variable for total	

2021

	(b)	(i)	• Number with a decimal / fractional part • Suitable example (e.g. 17.24)	2	One mark for definition, one mark for example Do not accept float as definition Allow fractions as example
		(ii)	• Whole number // number with no decimal / fractional part • Suitable example (e.g. 17)	2	One mark for definition, one mark for example

	(b)	(i)	• 16	1	
		(ii)	• 2	1	
		(iii)	• 9	1	
	(c)	(i)	• second.substring(3,5)	1	Ignore print / lack of print. Allow other suitable methods of string manipulation as long as variables used. Allow any valid method that extracts rightmost 5 or 6 characters of `second` variable.
		(ii)	• first.substring(0,8)	1	Ignore print / lack of print. Allow other suitable methods of string manipulation as long as variables used. Allow any valid method that extracts leftmost 8 or 9 characters of `first` variable.
		(iii)	• first.substring(9,7) + " " + second • "Science " + second • first.substring(9,7) + " is great"	1	Ignore print / lack of print. Allow other suitable methods of string manipulation as long as variables(s) used. Allow alternative concatenation symbols (e.g. & or .). Allow concatenation functions Must have correct spacing in outcome.

<table>
<tr><td>6</td><td>(a)</td><td></td><td><table><tr><th>Function call</th><th>Returned value</th></tr><tr><td>checkblock(2,1)</td><td>B</td></tr><tr><td>checkblock(3,0)</td><td>A</td></tr><tr><td>checkblock(2,3)</td><td>FREE</td></tr></table></td><td>3</td><td>Do not accept “blank” or any other returned value for third call.

Ignore case and spelling as long as recognisable.</td></tr>
<tr><td></td><td>(b)</td><td></td><td>• Returns a value // passes back a value</td><td>1</td><td></td></tr>
<tr><td></td><td>(c)</td><td>(i)</td><td>• Parameter values outside index range / larger than 4 / smaller than 0 // -1, 16 is not a valid block</td><td>1</td><td>Answer must refer to either array or gameboard / grid / block</td></tr>
<tr><td></td><td></td><td>(ii)</td><td>• Use selection / IF / Switch-Case / range check
• …check that parameters are >=0 and <= 4…
• …Return error code if invalid // set outcome to error</td><td>3</td><td>Allow equivalent checks (e.g. <5, between 0 and 4) for BP2
Allow reference to <code>r</code> and <code>c</code> as parameters.
BOD handle error for BP3 (e.g. repeat until valid)
Answer must be a description, code by itself is NAQ</td></tr>
<tr><td></td><td>(d)</td><td></td><td>• Input two position values separately
• calls <code>checkblock()</code> function…
• …with input parameters
• … returned value used in selection
• If free, stores “A” to correct index of <code>gamegrid</code> array (FT for incorrect selection)
• Loops until free position chosen</td><td>6</td><td>If flowchart / structured English, do not allow simple repeat of question.
<u>Example answer</u>
<pre>loop = True
while loop
    row = input("enter row")
    col = input("enter column")
    if checkblock(row,col) == "FREE" then
        gamegrid[row,col] = "A"
        loop = False
    endif
endwhile</pre></td></tr>
</table>

2020

<table>
<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>3</td><td>(b)</td><td>(i)</td><td>• <code>money</code>
• <code>price</code></td><td>1

AO1
1b(1)</td><td>Must be an identifier, not description. Ignore case.</td></tr>
<tr><td>3</td><td>(b)</td><td>(ii)</td><td>• one</td><td>1

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

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	(d)	(i)	• 3	1 AO1 1b(1)	CAO
4	(d)	(ii)	• 1	1 AO1 1b(1)	CAO

6	(c)		• Use of iteration (any use) …
			• ...loops for each item in array // loops 6 times
			• …to print out each item in studentnames
			• …input attendance
			• Add up/calculate students present **and** absent
			• …Outputs present **and** absent (in suitable message)
				6
				
				AO3
				2b(6)
					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>
					
					`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)`

2019

4	(c)		1 mark per bullet to max 3 • `count` • `= nogoalscount + 1` • `nogoalscount`	3 AO3 2b (3)	Correct answer only. Accept alternatives to adding 1 to variable (e.g. `+= 1` / `++`) Penalise spelling once only, FT for further mistakes. Do not penalise case. Accept sensible messages printed out alongside `nogoalscount`
5	(c)		• 9	1 AO1 1b (1)	Correct answer only Do not accept 3^2 or 3 x 3

6	(a)	(i)	1 mark per bullet to max 6
			
			• Function ticketprice() **defined**
			• … that accepts two parameters and has no other inputs
			• Works out total ticket price for adult (eg adult * 19.99)
			• Works out total ticket price for children (eg child * 8.99)
			• Adds on **correct** booking fee
			• Returns the calculated value.
				6
				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.
					
					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.
					
					Allow 2.50 booking fee to be per order or per ticket
					
					Ticket prices must be stored appropriately if needed.
					
					example algorithm
					
					`function ticketprice(numadult, numchild)`
					`  price = (numadult * 19.99) + (numchild * 8.99) + 2.50`
					`  return price`
					`end function`
					
					Allow alternatives in high level languages (e.g. def in Python).
					
					Allow return as assigning the value to the name of the function (VB syntax)

6	(a)	(ii)	1 mark per bullet to max 2 • Real… • …Returned value may not be a whole number / may have a decimal point in	2 AO2 1a (1) AO2 1b (1)	Allow String only if matching justification shows understanding (e.g. £ sign attached, message returned alongside value).

2018

1	(a)		1 mark per bullet, max 3 • String • Integer / Int • Boolean	3	Accept text / varchar for string. Do not accept character. Do not accept number/numeric for integer Accept yes/no, true/false for Boolean.
1	(b)	(i)	1 mark per bullet, max 2 if not in correct order or additional statements given. • SELECT StudentName • FROM conduct • WHERE Points < 0	3	Capitalisation does not affect the mark. Spellings of fields, tables must be correct. Ignore brackets. Ignore quotes around StudentName, Conduct or Points. Mark quotes around 0 in WHERE clause as incorrect. StudentName must not include space Accept <= -1 or equivalent for 3rd bullet point.
1	(b)	(ii)	• * / star / asterisk	1	Wildcard (*) must be clearly identified as the answer. Do not allow any other SQL statements alongside this unless this is given as an example.

1	(c)		1 mark per bullet, max 4
			
			• Selection(IF) used
			• Comparing studentdata[3]…
			• …with "TRUE" or "FALSE" // TRUE or FALSE
			• Correct outputs ("sent" **and** "not sent")
				4	Example algorithm
					
					`if studentdata[3] == "TRUE" then`
					`  print "sent"`
					`else`
					`  print "not sent"`
					`end if`
					
					Bullet point 3 can only be awarded If an attempt is made at identifying `studentdata` (e.g. with the wrong index or no index). Do not allow simply comparing anything with True / False.
					
					Bullet point 3 can be implicit.

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 • Sequence • Iteration / loops / repetition	2	Ignore spelling. Do not allow examples (eg FOR loop / WHILE loop)
2	**(c)**	**(i)**	1 mark per bullet, max 2 • A (name/identifier for a) **memory location** • used to (temporarily) **holds/contains/stores** data / value // is assigned a value • that **can be** changed / **possible** to change (while the program is running)	2	Do not accept "will change" for bullet point 4. Do not allow "holds/stores something" or "holds/stores information" for bullet point 2. 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 • k • p • m	2	Ignore capitalisation. Correct answer only. Do not allow other code in answer.

4	**(a)**	**(i)**	1 mark per filled gap, max 3
			`01 function librarycode(title, year)`
			`02 parta = title.substring(0, 3)`
			`03 partb = year.substring(2, 2)`
			`04 return parta.upper + partb`
			`05 endfunction`
				3	Ignore capitalisation.
					Allow **librarycode =** for 3rd mark – this is an equivalent in some languages for returning a value (eg. Visual Basic).

4	**(a)**	**(ii)**	1 mark per bullet, max 6
			• Input title and year from user
			• Open bookcodes.txt
			• Call the librarycode() function…
			• … with the two parameters that match input values
			• … write out **code obtained** to the text file
			• Close text file
				6	Example algorithm
					`title = input("enter title")`
					`year = input ("enter year")`
					`code = librarycode(title, year)`
					`myFile = openWrite("bookcodes.txt")`
					`myFile.writeLine(code)`
					`MyFile.close()`
					Note, pseudocode shown above is an example – candidates may answer very differently, but award marks if intention can be seen.
					Bullet points 3,4 and 5 could be done in one line:
					`myFile.writeLine(librarycode(title, year))`
					Do not award bullet point 3 if candidate is defining the function rather than calling it.
					Allow bullet point 2 (opening text file) if correctly referred to during write operation.
					Bullet point 3 must include brackets () to signify it is the function being called or indication that is being called.

4	**(b)**	**(i)**	1 mark per bullet, max 2. • Function **returns** a value • Procedure **does not return** a value	2	Allow "does not" for second mark if intention is clear (ie if it is obvious that the "not" refers to not returning a value). Allow discussion of how returned value in a function can be used (e.g. to assign to a variable or to use this returned value in some way).

<table>
<tr>
<td>4</td><td>(b)</td><td>(ii)</td>
<td>1 mark per bullet, max 4. Mark in pairs.

e.g.
• Breaks down / decomposes / modularises the problem / program // structures the program
• …making it easier to design/create/test
• …each subroutine can be tested separately

• Reuse code (in different programs)
• …quicker to develop (new) programs
• …build on existing work / use of a library of subroutines

• Avoid repetition of code (in the same program)
• …makes program shorter / smaller
• … subprogram called instead of copying/pasting.
• … quicker to develop (new) programs

• Easier to maintain
…as code is easier to understand/read
• …as code is shorter

• Easier to debug
• …as code is shorter
• …same bugs will not have been copied to other areas of the program.

• Work can be split up in a team
• ...to suit developers' skill set
• …to work on different subprogram at the same time / develop separately

• Allows for abstraction / removes complexity
• …subprograms can be used by programmers who do not need to understand how they work.</td>
<td>4</td>
<td>Maximum of two benefits with expanstions to be marked as per question.

Allow other sensible expansions.

Allow expansions which cross over to other benefits (e.g. breaks down the problem / to make it easier to maintain).

Allow “can be called multiple times”

Allow “file size is smaller”.

Do not allow “more efficient” without further explanation.</td>
</tr>
<tr>
<td>6</td><td>(a)</td><td></td>
<td><table><tr><th>Will loop infinitely</th><th>Will <u>not</u> loop infinitely</th></tr><tr><td>✓</td><td></td></tr><tr><td>✓</td><td></td></tr><tr><td></td><td>✓</td></tr><tr><td></td><td>✓</td></tr></table></td>
<td>4</td>
<td>1 mark per row.

More than one tick in a row = 0 marks for that row.</td>
</tr>
<tr>
<td>6</td><td>(b)</td><td></td>
<td>1 mark per bullet, max 3.

• FOR loop used
• That outputs the counter variable
• loops 10 time</td>
<td>3</td>
<td>Example algorithm

<pre>for i = 1 to 10
    print i
next</pre>
Do not accept WHILE loop for first mark, although other marks can be accessed.

No need for <code>next</code>

If candidate manually increments counter within FOR loop, do not award bullet point 3.

Accept pseudocode that suggests looping 10 times, even if this may not function correctly in a specific language.</td>
</tr>
<tr>
<td>7</td><td>(a)</td><td>(i)</td>
<td>1 mark per bullet, max 2.

• <code>else</code>
• <code>print (“unknown”)</code></td>
<td>2</td>
<td>Accept logically correct equivalents for else (e.g. <code>elseif a!=”LAN” and/or a !=”WAN”)</code>. Do not allow <code>elseif</code> on its own

Accept other keywords for print (e.g. “output”) as long as the intention is clear.

Accept other messages as equivalent to “uknown” (e.g. “not known “ / “error”))

Message to be printed must be in quotes.

Allow “<code>else then</code>”.</td>
</tr>
</table>

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".

2016

4	a		Sequence	1	
4	b		• A location in memory • The value/contents **cannot** be changed (whilst the program is running)	2	0 mark for "a variable that does not change" 0 marks for "stays the same"
4	c		numberOfPages = numberOfPages+numberOfChapters	1	Accept: • += instead of = numberOfPages • numberOfPages=RoundDown(numberOfWords/wordsPerPage) +numberOfChapters • numberOfPages=RoundDown(numberOfWords/300) +numberOfChapters Variable names must be spelt correctly, ignore case
4	d		• Integer/Int • It is a whole number/you can't have half a word	2	Do not allow 'need to ignore the decimal' Cannot get reason if data type incorrect
4	e		• String (name) • Real/Single/Double/Currency/Float/(Decimal) (price)	2	

9			1 mark per bullet
			• Taking the move as input
			• Checking if array element input is free …
			o …Outputting if it is taken
			• Writing "A" to the correct array element
			• Counting how many free space there are…
			o …Outputting the number of free spaces (if good attempt at counting free spaces)
			e.g.
			`INPUT move`
			`IF numbers(move) = "" then`
			`  numbers(move) = "A"`
			`ELSE`
			`  output "taken"`
			`ENDIF`
			`free = 0`
			`FOR x = 0 TO 100`
			`  IF numbers(x) = "" then`
			`  free = free + 1`
			`  ENDIF`
			`NEXT x`
			`OUTPUT free`
			e.g.
			`INPUT move`
			`IF numbers(move) = "" then`
			`  numbers(move) = "A"`
			`  numberFree = numberFree - 1`
			`ELSE`
			`  output "taken"`
			`ENDIF`
			`OUTPUT numberfree`
				6	The output mark can only be awarded if a reasonable attempt at adding the free spaces have been performed
					
					Counting how many free spaces there are can be done by either:
					• Looping through each element of the array and updating a variable if free/taken
					• Subtracting 1 each time an element is taken (this must work, i.e. there is no initialisation of the variable e.g. to 101, as that would run every time and reset the variable). If Initialisation is used, this must be outside a loop and must be 101.

2015

10	a		• Lidia	1	Accept incorrect spelling if intention is clear.
	b		• Program finds there is no position 7 in the array / array index out of bounds • An error will occur / an error message would be displayed / program will crash	2	Only award bullet 1 if answer is clearly about the contents of the array and not about the context. Do not award bullet 2 if candidate specifically mentions syntax error.

	c		Example
			`INPUT Num`
			
			`For i = 1 to Num`
			`  Temp = PlayerName(6)`
			`  PlayerName(6) = PlayerName(5)`
			`  PlayerName(5) = PlayerName(4)`
			`  PlayerName(4) = PlayerName(3)`
			`  PlayerName(3) = PlayerName(2)`
			`  PlayerName(2) = PlayerName(1)`
			`  PlayerName(1) = Temp`
			`Next i`
			
			Award marks for:
			• Input the number of places to move (e.g. Num)
			• Use of temporary variable(s) or second array to avoid overwriting values in the array
			• Sensible use of a loop
			• ... with correct end condition
			• Correctly deals with moving from position 1 (e.g. 1 + Num)
			• Correctly deals with moving from position 6 (e.g. Num)
				6	If there is more than one loop, award bullets 3 and 4 for any non-trivial loop that contributes to the solution.
					
					For bullet 3, "sensible" use of a loop, requires that the loop clearly address the problem (e.g. move every player from pos a to b). Although candidates can get partial marks here, candidates will only get full marks (incl bullet 6) if all conditions of all loops are correct.

Extra Questions

Lists attributes Surname, Title, PhoneNo from the table CUSTOMER for all customers in Coventry e.g. for local promotions / new store opening

2		i	• (A description of) an item of data • That is passed to a subroutine (when it is called)… • …is used as a variable within the subroutine	2	**Examiner's Comments** The wording used by some candidates did not make it clear that the parameter was passed to the subroutine. Words such as 'fed', 'allocated', 'put', etc. were used. Few mentioned that the parameter was used as a variable within the subroutine.

3	a		• (WHILE) loop (on line 03) will repeat lines 04 to 08 or 03 to 09 • as long as n > 0 / until n is not > 0	2	**Examiner's Comments** Many good answers but a large minority tried to explain the principle of a while loop not how it was used in this algorithm.

5		i	• Defined within one module… • … accessible only in that module / Any mention of scope • Can be used as parameters • Data is lost at end of module • Same variable name can be used in other modules without overwriting values/causing errors • Can overwrite global variables (with the same name)	4	For module allow procedure / function / sub routine / block of code **Examiner's Comments** Well answered by most candidates.
		ii	• Defined at start of program • Exists throughout program / in all modules • Allows data to be shared by modules	2	**Examiner's Comments** Nearly all candidates were able to get at least one mark on this.

13	a		Iteration [1]	1 AO2.1 (1)	**Examiner's Comment:** Well answered by most candidates.
	b		It does not return a value [1]	1 AO2.1 (1)	**Examiner's Comment:** A number of candidates clearly did not appreciate how functions differ from procedures.
8			• Selection / Branching (1) (AO1.1) • Working selection example (1) (AO1.2) **e.g.** ``` if a>b then c=b+42 endif ``` • Iteration (1) (AO1.1) • Working iteration example (1) (AO1.2) **e.g.** ``` for count=1 to 10 print(count) next count ``` • Sequence (1) (AO1.1) • Working Sequence example (1) (AO1.2) **e.g.** ``` qty = input() total = qty * price ```	6	Max 6 marks Do not penalise pseudocode if it is does not conform to the specification pseudocode guidelines. **Examiner's Comments** The programming constructs of sequence, iteration and branching are specifically identified within the specification. Many candidates were unaware of these named constructs. Of those who were, many then failed to give a working example as required by the question, but went on to describe rather than exemplify. Reponses such as looping were too vague as candidates are expected to know the correct technical vocabulary at AS Level.

18			• Global variable is visible throughout a program / may be accessed from more than one part of the program (1), local variable is visible only in module / construct where it is created / declared (1).	2	Up to 2 marks for a valid description.
			Total	**2**	
19			• A function is a named section of program (1) that performs a specific task (1). • It returns a value (1), it is often called inline (1).	2	Up to 2 marks for a valid description.
			Total	**2**	
20			1 mark per bullet, max 2 for each tools Breakpoints • Use to test the program works up to/at specific points • Check variable contents at specific points • Can set a point where the program stops running Stepping • Can set the program to run line by line • Slow down/watch execution • Find the point where an error occurs	4	