



Practice Paper

GCSE (9-1) Computer Science
J277/02 Computational thinking, algorithms and programming

MARK SCHEME

Duration: 1 hour 30 minutes

MAXIMUM MARK 80

Version:
Last updated: 16/6/20
(FOR OFFICE USE ONLY)

MARKING INSTRUCTIONS

PREPARATION FOR MARKING














SCORIS

1. Make sure that you have accessed and completed the relevant training packages for on–screen marking: *scoris assessor Online Training*; *OCR Essential Guide to Marking*.
2. Make sure that you have read and understood the mark scheme and the question paper for this unit. These are posted on the RM Cambridge Assessment Support Portal <http://www.rm.com/support/ca>
3. Log–in to scoris and mark the **required number** of practice responses (“scripts”) and the **required number** of standardisation responses.

YOU MUST MARK 10 PRACTICE AND 10 STANDARDISATION RESPONSES BEFORE YOU CAN BE APPROVED TO MARK LIVE SCRIPTS.

	Assessment Objective
AO1	Demonstrate knowledge and understanding of the key concepts and principles of computer science.
AO1 1a	Demonstrate knowledge of the key concepts and principles of computer science.
AO1 1b	Demonstrate understanding of the key concepts and principles of computer science.
AO2	Apply knowledge and understanding of key concepts and principles of computer science.
AO2 1a	Apply knowledge of key concepts and principles of computer science.
AO2 1b	Apply understanding of key concepts and principles of computer science.
AO3	Analyse problems in computational terms: <ul style="list-style-type: none"> • to make reasoned judgements • to design, program, evaluate and refine solutions.
AO3 1	To make reasoned judgements (this strand is a single element).
AO3 2a	Design solutions.
AO3 2b	Program solutions.
AO3 2c	Evaluate and refine solutions.

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

COMPONENT 2 SECTION B SYNTAX GUIDANCE

In Section B, certain questions require candidates to answer in either the OCR Exam Reference Language or the high-level programming language they are familiar with. The information in this section provides generic guidelines in relation to the marking of these questions.

Where a response requires an answer in OCR Exam Reference Language or a high-level programming language, a candidate's level of precision will be assessed. These questions are designed to test both a candidate's programming logic and understanding of core programming structures.

Marks will be given for correctly using syntax to represent core programming constructs which are common across all programming languages. The construct must be present in a recognisable format in a candidate's answer.

Where the response requires a candidate to respond using the OCR Exam Reference Language or a high-level programming language, answers written in pseudocode, natural English or bullet points **must not** be awarded marks.

The guidance below covers the elements of each core construct. As guidance, several examples are provided for each. These examples are not exclusive but do present a variety of acceptable ways taken from a range of different languages.

Concept		Examiner Guidance
Commenting		
<pre>// //This function squares a number function squared(number) squared = number^2 return squared endfunction //End of function</pre>	<ul style="list-style-type: none"> • Other examples allowable, e.g.: <ul style="list-style-type: none"> ○ # this is a comment ○ /* this is another comment */ 	
Variables		
<pre>= const global x = 3 name = "Louise" const vat = 0.2 global userID = "Cust001"</pre>	<ul style="list-style-type: none"> • Variables and constants are assigned using the = operator • Constants are assigned using the <code>const</code> keyword (or similar) • Identifiers should not have clear spaces within them or start with numbers • String values must use quotation marks (or equivalent) • Assignment must use =, :=, ← (or a suitable alternative) • variable identifier must be on the left when using OCR Exam Reference Language and the value to be assigned on the right • Some languages allow the value on the left- and the identifier on the right-hand side • Variables and constants are declared the first time a value is assigned. They assume the data type of the value they are given • Variables and constants that are declared inside a function or procedure are local to that subroutine • Variables in the main program can be made global with the keyword <code>global</code> • For input, a suitable command word for input and a variable identifier to assign data to (if required) <p>e.g.</p> <pre>INPUT identifier identifier = INPUT</pre> 	

Input/Output		
input(...) print(...)	myName = input("Please enter a name") print("My name is Noni") print(myArray[2,3])	<ul style="list-style-type: none"> • For output, a command word for output (e.g. output, print, cout) • Data to be output. If this is a string then quotation marks (or equivalent) are required • If multiple items are to output, a suitable symbol for concatenation such as +, &.
Casting		
str() int() real() bool()	str(345) int("3") real("4.52") bool("True")	<ul style="list-style-type: none"> • Variables can be typecast using the int str and float functions
Iteration		
for ... to ... next ... for ... to ... step ... next ...	<pre>for i=0 to 9 print("Loop") next i for i=2 to 10 step 2 print(i) next i for i=10 to 0 step -1 print(i) next i</pre>	<ul style="list-style-type: none"> • for keyword • ...with counter variable • Identification of number of times to iterate • Clear identification of which section of code will be repeated (e.g. using indentation, next keyword or equivalent, {braces})
while ... endwhile do until ...	<pre>while answer != "Correct" answer = input("New answer") endwhile do answer = input("New answer") until answer == "Correct"</pre>	<ul style="list-style-type: none"> • While / do..until key words or equivalent • ...with logical comparison • clear identification of which section of code will be repeated (e.g. using indentation, endwhile/until keyword or equivalent, braces)

Selection		
<pre>if ... then elseif ... then else endif</pre>	<pre>if answer == "Yes" then print("Correct") elseif answer == "No" then print("Wrong") else print("Error") endif</pre>	<ul style="list-style-type: none"> • if key word followed by logical comparison • key word for <code>elseif</code> or equivalent followed by logical comparison • key word for <code>else</code> or equivalent with no comparison • clear identification of which section of code will be executed depending upon decision
<pre>switch ... : case ... : case ... : default: endswitch</pre>	<pre>switch day : case "Sat": print("Saturday") case "Sun": print("Sunday") default: print("Weekday") endswitch</pre>	<ul style="list-style-type: none"> • May be referred to differently in some languages. The format to the left will be used in all questions • <code>switch/select</code> key word or equivalent followed by variable/ value being checked • key word for each case followed by variable/ value to compare to • key word for default case (last option) • clear identification of which section of code will be executed depending upon decision

String handling/operations		
<p>.length</p> <p>.substring(x , i)</p> <p>.left(i)</p> <p>.right(i)</p> <p>+ (concatenation)</p> <p>.upper</p> <p>.lower</p> <p>ASC (...)</p> <p>CHR (...)</p>	<pre>subject = "ComputerScience" subject.length gives the value 15</pre> <pre>subject.substring(3,5) returns "puter" subject.left(4) returns "Comp" subject.right(3) returns "nce"</pre> <pre>print(stringA + string) print("Hello, your name is : " + name)</pre> <pre>subject.upper gives "COMPUTERSCIENCE" subject.lower gives "computerscience"</pre> <pre>ASC(A) returns 65 (numerical) CHR(97) returns 'a' (char)</pre>	<ul style="list-style-type: none"> • Suitable key word to indicate length and string identifier e.g. len(string) • Suitable string and characters required identified • Use of key words such as left, right, mid, etc, are all acceptable as long as these are precise • Treating a string as an array of characters is acceptable • Alternate symbol used indicate two strings or values are being concatenated is acceptable e.g. <code>stringA & stringB</code> or <code>stringA.stringB</code> • Use of comma e.g. <code>print(stringA, stringB)</code> is acceptable to output multiple values but examiners should be aware that this is not concatenation. • Suitable key word to indicate string to be converted and whether this is to be converted to upper or lower case e.g. lower(stringname) • Suitable keyword to indicate conversion and whether this is to or from ASCII. Where converting from ASCII, an integer value must be given and where converting to ASCII, a single character must be given.

File handling		
<pre>open(...) .close() .readLine() .writeLine(..) .endOfFile() newFile()</pre>	<pre>myFile = open("sample.txt") myFile.close() myFile.readLine() returns the next line in the file myFile.writeLine("Add new line") while NOT myFile.endOfFile() print(myFile.readLine()) endwhile newFile("myText.txt")</pre>	<ul style="list-style-type: none"> • open keyword (or equivalent) • read or write clearly identified • write or read keyword (or equivalent) • close file keyword (or equivalent) • newFile keyword (or equivalent)
Arrays		
<pre>array colours[...] array gameboard[...,...] names[...] = ... gameboard[...,...] = ...</pre>	<pre>array colours[5] array colours = ["Blue", "Pink", "Green", "Yellow", "Red"] array gameboard[8,8] names[3] = "Noni" gameboard[1,0] = "Pawn"</pre>	<ul style="list-style-type: none"> • Array identifier • Index number to be accessed in square brackets, rounded brackets or curly braces (all acceptable) • Array identifier assigned to initial values in one step • For 2D arrays, the two indices should be given in one bracket separated by a comma or in two separate brackets, e.g. gameboard[4,6] gameboard[4][6] <p>Where 2D arrays are represented by tables in a question, candidates are expected to use the same row/column or column/row format as given in the question. This will always be given.</p>

Sub programs

<pre>procedure name (...)</pre>	<pre>procedure agePass() print("You are old enough to ride") endprocedure procedure printName(name) print(name) endprocedure procedure multiply (num1, num2) print(num1 * num2) endprocedure</pre>	<ul style="list-style-type: none"> • function or procedure key word (or equivalent) • ... followed by identifier • Any parameters passed in are contained within brackets and come after identifier name • Clear identification of which section of code is contained within the subroutine (e.g. indentation, endsub key word, braces)
<pre>procedure (parameters)</pre>	<pre>agePass () printName (parameter) multiply (parameter1, parameter2)</pre>	<ul style="list-style-type: none"> • functions only: a suitable method of returning a value (e.g. <code>return</code> keyword or assignment of value to function identifier)
<pre>function name (...)</pre> <pre> ...</pre> <pre> return ...</pre> <pre>endfunction</pre>	<pre>function squared(number) squared = number^2 return squared endfunction</pre>	<p>e.g.</p> <pre>def newfunction(x,y) total = x + y newfunction = total</pre>
<pre>function (parameters)</pre>	<pre>print (squared(4)) newValue = squared(4)</pre>	

Random numbers		
random(..., ...)	myVariable = random(1, 6)	<ul style="list-style-type: none"> • random key word (or equivalent) • identification of either smallest and largest number to be chosen or just largest number <p>e.g. randnumber (10) rand (1 , 6)</p>
	myVariable = random(-1.0, 10.0)	

Comparison operators			
==	Equal to	<=	Less than or equal to
!=	Not equal to	>	Greater than
<	Less than	>=	Greater than or equal to
Boolean operators			
AND	Logical AND		
OR	Logical OR		
NOT	Logical NOT		
Arithmetic operators			
+	Addition		
-	Subtraction		
*	Multiplication		
^	Exponent		
/	Division		
MOD	Modulus		
DIV	Quotient		

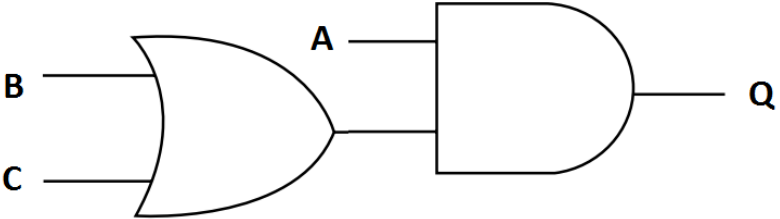
- = or == are both acceptable for equal to.
- <> is acceptable for not equal to.
- Care must be taken by candidates to ensure that > and < are not mixed up.
- Candidates must understand that < and > are non-inclusive, so that <9 does not include 9. This is different than <=9 which is inclusive and therefore does include 9.
- Alternative symbols for arithmetic operators are acceptable where these appear in other high-level languages (such as % for MOD or ** for exponentiation).
- 6 x 5 is not an acceptable alternative for multiplication.
- Alternative logical operators are acceptable where these appear in other high-level languages (such as && for **AND**).
- Alternative Arithmetic Operators may be used as well (such as % for modulus).
- Candidates must be aware that logical operators must be used correctly:

if x > 0 AND x < 10 is logically correct.
if x > 0 AND < 10 is **not** logically correct.

Question			Answer	Mark	Guidance															
1	a	i	One mark per row	4	Accept other markings that indicate a choice has been made (e.g. a cross, etc)															
			<table border="1"> <thead> <tr> <th>Statement</th> <th>High-level language</th> <th>Low-level language</th> </tr> </thead> <tbody> <tr> <td>Uses English-like keywords such as <code>print</code> and <code>while</code>.</td> <td style="text-align: center;">✓</td> <td></td> </tr> <tr> <td>Must be translated before the processor can execute code.</td> <td style="text-align: center;">✓</td> <td></td> </tr> <tr> <td>Code written is portable between different processors</td> <td style="text-align: center;">✓</td> <td></td> </tr> <tr> <td>Requires the programmer to understand the processor's registers and structure</td> <td></td> <td style="text-align: center;">✓</td> </tr> </tbody> </table>			Statement	High-level language	Low-level language	Uses English-like keywords such as <code>print</code> and <code>while</code> .	✓		Must be translated before the processor can execute code.	✓		Code written is portable between different processors	✓		Requires the programmer to understand the processor's registers and structure		✓
			Statement			High-level language	Low-level language													
			Uses English-like keywords such as <code>print</code> and <code>while</code> .			✓														
			Must be translated before the processor can execute code.			✓														
Code written is portable between different processors	✓																			
Requires the programmer to understand the processor's registers and structure		✓																		
	b		1 mark per bullet, max 4	4	Allow other tools available in an IDE with suitable expansion (e.g. breakpoints, watch window, stepping, pretty printing, etc)															
			e.g.																	
			<ul style="list-style-type: none"> • Editor • ...to enable program code to be entered / edited 																	
			<ul style="list-style-type: none"> • Error diagnostics / debugger • ...to display information about errors / location of errors / suggest solutions 																	
			<ul style="list-style-type: none"> • Run-time environment • ...to enable program to be run / to check for run-time errors / test the program 																	

Question		Answer	Mark	Guidance	
2	a	<p>1 mark per bullet, max 4</p> <ul style="list-style-type: none"> • C • A • D/F • F/D 	4	<p>D, F may be swapped around.</p> <p>e.g.</p> <pre> graph TD Root[Mobile phone app] --> Login[Login] Root --> Manage[Manage appointments] Root --> C[C] Manage --> A[A] Manage --> View[View appointments] View --> DF[D / F] View --> FD[F / D] </pre>	
	b	i	<ul style="list-style-type: none"> • An error that does not cause the program to crash // produces unexpected output 	1	
		ii	<p>1 mark per bullet, max 4</p> <ul style="list-style-type: none"> • Line 02 // empty = 0 • Will reset empty to 0 on each iteration of the loop • Line 07 // print ("empty") • Will print out the string "empty" instead of the value held in the variable 	4	Mark in pairs
	c	i	<p>1 mark per bullet, max 4</p> <ul style="list-style-type: none"> • Compare 5 (middle value) to 7 • 5 is smaller than 7 / 7 is larger than 7 so... • discard lower part of list / repeat with upper part of list • ...compare 7 to 7 (item found) 	4	Do not accept generic answers that do not refer to the data given.

		ii	1 mark per bullet, max 2 <ul style="list-style-type: none"> List of size 1 to compare ...and item not matched to search term 	2	Do not accept answers relating to "end of list" – this is linear search.
		iii	<ul style="list-style-type: none"> More efficient // Less time taken (to find item) // fewer comparisons to make (with large lists) 	1	Accept reference to big O notation as equivalent to more efficient.

Question		Answer	Mark	Guidance
3	a	<ul style="list-style-type: none"> OR gate with two inputs // AND gate with two inputs Diagram as shown in guidance with no additional gates 	2	
	b	<ul style="list-style-type: none"> Logically compares A AND // correct nested IF ...B OR C // correct sequential IF Output in both cases (with attempt at selection). 	3	<pre>A = input("Is the customer 15 or over?") B = input("Does the customer have a ticket?") C = input("Does the customer money to buy a ticket") if A AND (B OR C) then print ("allowed") else print ("not allowed") endif</pre> <p>Accept answers where inputs are given as strings e.g :</p> <pre>if A == "Yes" AND (B == "Yes" OR C == "Yes") then print ("allowed") else print ("not allowed") endif</pre>
	c	<ul style="list-style-type: none"> freeseats called with "Red" ...<u>returned value</u> assigned to variable <u>redseats</u> 	2	<pre>redseats = freeseats("Red")</pre>

					"Red" must use suitable string delimiters (e.g. speech marks) if directly passing the string. Do not penalise case.
--	--	--	--	--	---

Question			Answer	Mark	Guidance												
4	a	i	<ul style="list-style-type: none"> Hiding / ignoring / removing detail // focussing on certain parts of a problem 	1													
		ii	<ul style="list-style-type: none"> Focus on age / number of miles Ignore other factors (such as make, model, etc) 	1	Allow other examples of factors to ignore / remove for BP2												
		iii	<ul style="list-style-type: none"> Ensures only certain users can access the system Using password / other example of authentication technique 	2	Allow other examples of authentication for BP2												
	b	i	<p>1 mark per bullet, max 4</p> <ul style="list-style-type: none"> Miles and age input <u>separately</u> Checks for valid mileage Checks for valid age Checks <u>both</u> are greater than / greater than equal to zero ...correctly outputs both True and False 	5	<p>BP2 and 3 must check for both ends of range – must check that input data is not negative.</p> <p>Allow FT for BP4 if already penalised under BP2 and/or 3 and output is otherwise correct.</p> <p>e.g.</p> <pre>miles = input("enter miles driven") age = input("enter age of car") valid = True if miles > 10000 or miles < 0 then valid = False elseif age > 5 or age < 0 then valid = False endif print(valid)</pre>												
		ii	<p>1 mark per row, max 3</p> <ul style="list-style-type: none"> Normal : miles (0 – 9,999), age (0 - 5) Erroneous/Invalid: miles (less than 0, larger than 9,999), age (less than 0 / more than 5) // non-numeric data Boundary : miles (-1/0 / 9,999 / 10,000), age (-1/0 / 5/6) 	3	<p>Specific data must be given, not a description e.g.</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th></th> <th>Miles</th> <th>Age</th> </tr> </thead> <tbody> <tr> <td>Normal</td> <td>7,000</td> <td>3</td> </tr> <tr> <td>Erroneous</td> <td>12,000</td> <td>7</td> </tr> <tr> <td>Boundary</td> <td>10,000</td> <td>5</td> </tr> </tbody> </table>		Miles	Age	Normal	7,000	3	Erroneous	12,000	7	Boundary	10,000	5
	Miles	Age															
Normal	7,000	3															
Erroneous	12,000	7															
Boundary	10,000	5															

		iii	<ul style="list-style-type: none"> • During development // whilst writing the program // before development is complete. 	1	
	c		<p>1 mark per bullet, max 6</p> <ul style="list-style-type: none"> • Inputs the current battery charge percentage • Outputs "full" if 100% • Calculates the amount to charge • Calculates the time in minutes... • ...converts to hours and minutes • Outputs the time in hours and minutes 	6	<p>Allow output of 0 hours 0 minutes if full. Allow answers referencing decimal parts (e.g. 0.8 = 80%) BP5 can be attempted in many ways (e.g. DIV and MOD, repeated division, etc) Allow FT for BP6 if reasonable attempt at conversion for BP5 has been given.</p> <p>e.g. <pre>charge = input("enter battery charge") if charge == 100 then print("full") else time = (100-charge) * 10 hours = time DIV 60 mins = time MOD 60 print (hours, mins) endif</pre></p>

Question		Answer	Mark	Guidance																		
5	a	<p>One mark per correct choice</p> <ul style="list-style-type: none"> • SELECT ItemCode, ItemName • FROM tblStock • WHERE Price >=60 	3	Accept other markings that indicate a choice has been made (e.g. a cross, etc)																		
	b	<p>i</p> <p>One mark if two correct, two marks if four correct, three marks if all correct.</p> <table border="1" data-bbox="427 560 1113 826"> <thead> <tr> <th>Price input</th> <th>Test type</th> <th>Expected price output</th> </tr> </thead> <tbody> <tr> <td>50</td> <td>Normal</td> <td>50</td> </tr> <tr> <td>100</td> <td>Boundary</td> <td>100</td> </tr> <tr> <td>150</td> <td>Normal</td> <td>130</td> </tr> <tr> <td>200</td> <td>Boundary</td> <td>180</td> </tr> <tr> <td>250</td> <td>Normal</td> <td>210</td> </tr> </tbody> </table>	Price input	Test type	Expected price output	50	Normal	50	100	Boundary	100	150	Normal	130	200	Boundary	180	250	Normal	210	3	
Price input	Test type	Expected price output																				
50	Normal	50																				
100	Boundary	100																				
150	Normal	130																				
200	Boundary	180																				
250	Normal	210																				
		<p>ii</p> <p>One mark per bullet point</p> <ul style="list-style-type: none"> • Input and store price • Check if price is > 200... • ...if true, reduce price by 40 • Check if price is >100 <u>and not >200</u>... • ...if true, reduce price by 20 • Output price 	6	<p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural language.</p> <p>BP3 and BP5 only to be given if sensible check for price being over the appropriate threshold. BP4 must check that price is both larger than 100 and not larger than 200; do not give mark for simply checking price is larger than 100. This may be implicit (e.g. using elseif).</p> <p>e.g.</p> <pre>price = input("enter price") if price > 200 then price = price - 40 elseif price > 100 then price = price - 20 endif print(price)</pre>																		

	c	<p>One mark per bullet point</p> <ul style="list-style-type: none"> • checking both values (e.g. <code>or</code> changed to <code>and</code> if appropriate) • <code>if</code> statement in correct format (e.g. checking against <code>stocklevel</code> for each condition) • <code>if</code> statement uses correct comparisons (e.g. <code>>=</code> and <code><=</code>) • <code>print</code> statements in correct position • <code>print</code> statements include string delimiters (e.g. speech marks) around both string outputs 	5	<p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural language.</p> <p>e.g.</p> <pre>stocklevel = input("Enter stock level") if stocklevel >= 5 and stocklevel <= 25 then print("In demand") else print("Not in demand") endif</pre> <p>alternative example</p> <pre>stocklevel = input("Enter stock level") if stocklevel < 5 or stocklevel > 25 then print("Not in demand") else print("In demand") endif</pre> <p>As a matter of principle, a candidate who refines the program to work fully but in a different format to that specified should gain full marks.</p>
	d	<p>i</p> <p>One mark per bullet point, in the correct place</p> <ul style="list-style-type: none"> • <code>size // len(discountcodes-1)</code> • <code>code</code> • <code>price // newprice</code> • <code>[x,1] // [x][1]</code> • <code>return newprice // checkdiscount = newprice</code> 	5	<p>e.g.</p> <pre>function checkdiscount(price, code) newprice = price size = len(discount)-1 for x = 0 to size if discount[x,0] == code then newprice = price - discount[x,1] endif next return newprice endfunction</pre>

	d	ii	<p>One mark per bullet point, maximum 2 marks</p> <ul style="list-style-type: none"> • newprice • size • x 	2	<p>Do not penalise capitalisation</p> <p>Accept price, code, discount</p>
	d	iii	<ul style="list-style-type: none"> • asks for price and discount code to be input • ...passes both to the checkdiscount() function as parameters... • ...stores / uses returned value • calculates total of all prices entered/returned • repeats until 0 is entered as <u>price</u> • outputs <u>calculated total</u> 	6	<p><u>High-level programming language / OCR Exam Reference Language response required</u></p> <p>Do not accept pseudocode / natural language.</p> <p>BP3 allow total of prices entered as FT if candidate does not achieve BP2</p> <p>e.g.</p> <pre>total = 0 do price = input("Enter a price") code = input("Enter a discount code") newprice = checkdiscount(price, code) total = total + newprice until price == 0 print(total)</pre> <p>alternative example</p> <pre>total = 0 price = 1 while price != 0 price = input("Enter a price") code = input("Enter a discount code") total = total + checkdiscount(price, code) endwhile print(total)</pre>

J277/02

Mark Scheme

Practice paper

--	--	--	--	--	--