

# Cambridge International AS & A Level

---

**COMPUTER SCIENCE****9618/23**

Paper 2 Fundamental Problem-solving and Programming Skills

**October/November 2024**

MARK SCHEME

Maximum Mark: 75

---

**Published**

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

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

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

Cambridge International is publishing the mark schemes for the October/November 2024 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

---

This document consists of **14** printed pages.

**Generic Marking Principles**

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

**GENERIC MARKING PRINCIPLE 1:**

Marks must be awarded in line with:

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

**GENERIC MARKING PRINCIPLE 2:**

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

**GENERIC MARKING PRINCIPLE 3:**

Marks must be awarded **positively**:

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

**GENERIC MARKING PRINCIPLE 4:**

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

**GENERIC MARKING PRINCIPLE 5:**

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

**GENERIC MARKING PRINCIPLE 6:**

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

**Mark scheme abbreviations**

/	separates alternative words / phrases within a marking point
//	separates alternative answers within a marking point
<b>underline</b>	actual word given must be used by candidate (grammatical variants accepted)
<b>max</b>	indicates the maximum number of marks that can be awarded
( )	the word / phrase in brackets is not required, but sets the context

**Note:** No marks are awarded for using brand names of software packages or hardware.

Question	Answer				Marks
1(a)	Pseudocode example	Selection	Iteration	Subroutine	4
	FOR Index ← 1 TO 3 IF Safe[Index] = TRUE THEN Flap[Index] ← 0 ENDIF NEXT Index	✓	✓		
	CASE OF Compound(3)	✓		✓	
	REPEAT UNTIL AllDone() = TRUE		✓	✓	
	WHILE Result[3] <> FALSE		✓		
	One mark per row				
1(b)	Variable	Example data value	Data type		3
	Available	TRUE	BOOLEAN		
	Received	"18/04/2021"	STRING		
	Index	100	INTEGER		
	One mark per row				
1(c)	Expression		Evaluates to		3
	Available AND NOT(Index > 100)		TRUE		
	Index MOD 30		10		
	NUM_TO_STR(Index + "33")		ERROR		
	One mark per row				

Question	Answer	Marks
2(a)	<p>DECLARE Count, Total, NextNumber : INTEGER</p> <p>Total ← 0</p> <p>FOR Count ← 1 TO 100</p> <p>    OUTPUT "Input an integer value"</p> <p>    INPUT NextNumber</p> <p>    IF NextNumber &gt; 0 THEN</p> <p>        Total ← Total + NextNumber</p> <p>    ENDIF</p> <p>NEXT Count</p> <p>OUTPUT Total</p> <p>Mark as follows:</p> <p><b>MP1</b> Declarations of all variables used</p> <p><b>MP2</b> Loop for 100 iterations</p> <p><b>MP3</b> Prompt <b>and</b> input a value <b>in a loop and</b></p> <p><b>MP4</b> Test for value &gt; 0 // &gt;=1 <b>in a loop</b></p> <p><b>MP5</b> Sum the Total <b>in a loop</b></p> <p><b>MP6</b> Output of Total after the <b>loop</b></p> <p><b>Max 5 marks</b></p>	5
2(b)	<p><b>MP1</b> Construct: Iteration / Repetition</p> <p><b>MP2</b> Use: To loop through all <u>100</u> inputs // To loop <u>100</u> times</p> <p>ALTERNATIVE:</p> <p><b>MP1</b> Construct: Selection</p> <p><b>MP2</b> Use: To test whether the value input is positive</p>	2

Question	Answer	Marks
3(a)(i)	0 is not a valid <u>array</u> / <u>List index</u> value // No 0 <sup>th</sup> element in the array	1
3(a)(ii)	0 to 200	1

Question	Answer	Marks
3(b)	<div><pre>graph TD; START([START]) --&gt; SET1[SET Index TO 1]; SET1 --&gt; IS200{IS Index = 200?}; IS200 -- NO --&gt; SETPTR[SET List[Index].Pointer TO Index + 1]; SETPTR --&gt; SETINC[SET Index TO Index + 1]; SETINC --&gt; IS200; IS200 -- YES --&gt; SET0[SET List[Index].Pointer TO 0]; SET0 --&gt; END([END]);</pre></div> <p>Mark as follows:</p> <ul style="list-style-type: none"><li><b>MP1</b> Use of variable as array index <b>and</b> initialisation to 1 / to first element of array</li><li><b>MP2</b> Loop for 199 / 200 iterations</li><li><b>MP3</b> Assign Index+1 to each pointer field in a loop</li><li><b>MP4</b> Assign 0 to 200<sup>th</sup> pointer field</li></ul>	4

Question	Answer	Marks
3(c)	<p>One mark per step:</p> <p><b>MP1</b> Assign <code>HeadPointer</code> to <code>ThisPointer</code> / <code>Current Pointer</code> // Identify first node using <u>headpointer</u></p> <p><b>MP2</b> Loop the following until <code>ThisPointer</code> value is 0 (zero) / null <u>pointer</u> reached</p> <p><b>MP3</b> ... Output data (field) from array element at index <code>ThisPointer</code> // Output data (field) of the current node / array element</p> <p><b>MP4</b> ... Assign pointer field from current array element / index / to <code>ThisPointer</code> / <code>Current Pointer</code> // Assign pointer field from current node to <code>ThisPointer</code> / <code>Current Pointer</code></p>	4
4(a)	<p>One mark per highlighted part:</p> <pre> FOR Index ← 1 TO <u>5</u>   Lower ← GB[Index] - 2   Upper ← <u>GB[Index] + 2</u> // <u>Lower + 4</u>   IF Mark <u>&gt;= Lower</u> AND Mark <u>&lt;= Upper</u> THEN //IF Mark <u>&lt;= Upper</u> AND Mark <u>&gt;= Lower</u> THEN     OUTPUT "Check this paper"   ENDIF NEXT Index </pre>	4
4(b)(i)	<p><b>MP1</b> Use <code>Mark</code> as the <u>index</u> to the <code>Check</code> array // to specify an array element</p> <p><b>MP2</b> If value of indexed element is <code>TRUE</code>, then the paper will need to be checked</p>	2

Question	Answer	Marks
4(b)(ii)	<p><b>Example:solution</b></p> <pre> PROCEDURE GbInitialise()   DECLARE GBIndex, GBMark, ThisIndex : INTEGER    FOR ThisIndex ← 0 TO 75     Check[ThisIndex] ← FALSE // initialise all values   NEXT ThisIndex    FOR GBIndex ← 1 TO 5     GBMark ← GB[GBIndex]     FOR ThisIndex ← GBMark - 2 TO GBMark + 2       Check[ThisIndex] ← TRUE //Set GBMark ± 2 to                                 TRUE     NEXT ThisIndex   NEXT GBIndex ENDPROCEDURE </pre> <p>Mark as follows:</p> <p><b>MP1</b> Procedure heading and ending  <b>MP2</b> Initialise Check array to FALSE  <b>MP3</b> Loop through GB array  <b>MP4</b> Extract the GB mark  <b>MP5</b> Loop for 5 elements across GBMark ± 2 // Check if within GBMark ± 2  <b>MP6</b> Assign each element of Check array to TRUE</p> <p><b>ALTERNATIVE – Example using selection Version 1</b></p> <pre> FOR ThisIndex ← 0 TO 75   CASE ThisIndex     GB[1] - 2 TO GB[1] + 2 : Check[ThisIndex] ← TRUE     GB[2] - 2 TO GB[2] + 2 : Check[ThisIndex] ← TRUE     GB[3] - 2 TO GB[3] + 2 : Check[ThisIndex] ← TRUE     GB[4] - 2 TO GB[4] + 2 : Check[ThisIndex] ← TRUE     GB[5] - 2 TO GB[5] + 2 : Check[ThisIndex] ← TRUE   OTHERWISE: Check[ThisIndex] ← FALSE   ENDCASE NEXT ThisIndex </pre>	6

Question	Answer	Marks
4(b)(ii)	<p><b>ALTERNATIVE – Example using selection Version 2</b></p> <pre> DECLARE ThisIndex, GBIndex : INTEGER DECLARE Lower, Higher : INTEGER  FOR ThisIndex ← 0 TO 75   For GBIndex ← 1 TO 5     Lower ← GB[GBIndex] - 2     Higher ← GB[GBIndex] + 2     IF ThisIndex &gt;= Lower AND ThisIndex &lt;= Higher       THEN         Check[ThisIndex] ← TRUE       ELSE         Check[ThisIndex] ← FALSE       ENDIF   NEXT Index NEXT ThisIndex </pre> <p>Mark as follows:</p> <p><b>MP1</b> Procedure heading and ending</p> <p><b>MP2</b> Loop through <i>Check</i> array// loop from 0 to 75</p> <p><b>MP3</b> Attempt at using CASE / Selection structure with five clauses/ <b>five</b> checks for range</p> <p><b>MP4</b> Extract GB grade</p> <p><b>MP5</b> Compare <i>ThisIndex</i> with each element in GB array <math>\pm 2</math></p> <p><b>MP6</b> Assign each element of <i>Check</i> array to TRUE if in range <b>or</b> FALSE if not in range</p>	

Question	Answer	Marks										
5(a)	<table><tr><th>Activity</th><th>Name of life cycle stage</th></tr><tr><td>A compiler is be used.</td><td>Coding</td></tr><tr><td>A program that has been released for general use is modified</td><td>Maintenance</td></tr><tr><td>The dry run method is used.</td><td>Testing</td></tr><tr><td>The program structure is specified.</td><td>Design</td></tr></table> <p>One mark per row</p>	Activity	Name of life cycle stage	A compiler is be used.	Coding	A program that has been released for general use is modified	Maintenance	The dry run method is used.	Testing	The program structure is specified.	Design	4
Activity	Name of life cycle stage											
A compiler is be used.	Coding											
A program that has been released for general use is modified	Maintenance											
The dry run method is used.	Testing											
The program structure is specified.	Design											
5(b)	<p>One mark for method – two marks for the description</p> <p><b>MP1</b> Stub testing</p> <p><b>MP2</b> The modules <u>Test A()</u> and <u>Test B()</u> are replaced by dummy modules</p> <p><b>MP3</b> ... which return a known result // An output statement is displayed when called (to check it works) // gives expected output</p>	3										

Question	Answer	Marks
6	<p><b>Loop Solution</b> Example solution:</p> <pre> FUNCTION AdjustClock(ThisYear : INTEGER) RETURNS INTEGER   DECLARE ThisDayNumber, SundayCount : INTEGER   DECLARE ThisDate : DATE    ThisDayNumber ← 0   SundayCount ← 0   REPEAT     ThisDayNumber ← ThisDayNumber + 1     ThisDate ← SETDATE(ThisDayNumber, 3, ThisYear)     IF DAYINDEX(ThisDate) = 1 THEN       SundayCount ← SundayCount + 1     ENDIF   UNTIL SundayCount = 3   RETURN ThisDayNumber ENDFUNCTION </pre> <p>Mark as follows:</p> <p><b>MP1</b> Function heading, parameter, ending and return type  <b>MP2</b> Declare local integer variable that is used to create a date  <b>MP3</b> Loop until 3<sup>rd</sup> Sunday found  <b>MP4</b> Attempt to use both SETDATE() and DAYINDEX() in a loop  <b>MP5</b> Correctly generate value of type DATE using SETDATE() in a loop  <b>MP6</b> Test if value represents a Sunday using DAYINDEX() in a loop  <b>MP7</b> Increment Sunday count in a loop and initialised correctly before loop  <b>MP8</b> Return day number of third Sunday</p> <p><b>Max 7 marks</b></p> <p><b>Non-loop solution</b> Example solution:</p> <pre> FUNCTION AdjustClock(ThisYear : INTEGER) RETURNS INTEGER   DECLARE FirstDayIndex: INTEGER   DECLARE ThisDate : DATE    ThisDate ← SETDATE(1, 3, ThisYear)   FirstDayIndex ← DAYINDEX(ThisDate)   IF FirstDayIndex = 1 THEN     ThirdSunday ← FirstDayIndex + 14 //First day is                                      Sunday   ELSE     ThirdSunday ← 23 - FirstDayIndex // Other days   ENDIF   RETURN ThirdSunday ENDFUNCTION </pre>	7

Question	Answer	Marks
6	<p>Mark as follows:</p> <p><b>MP1</b> Function heading, parameter, ending and return type</p> <p><b>MP2</b> Declare local integer variable that is used to hold a day number</p> <p><b>MP4</b> Attempt to use both <code>SETDATE()</code> <b>and</b> <code>DAYINDEX()</code></p> <p><b>MP5</b> Correctly generate value of type <code>DATE</code> using <code>SETDATE()</code> for first of March / specific day in March</p> <p><b>MP6</b> Test if date represents a Sunday / specific day using <code>DAYINDEX()</code> and calculate third Sunday // Correctly calculate third Sunday for a value returned by <code>DAYINDEX()</code> for one day</p> <p><b>MP7</b> Calculate third Sunday for other six days</p> <p><b>MP8</b> Return day number of <b>third</b> Sunday</p> <p><b>Max 7 marks</b></p>	

Question	Answer	Marks
7(a)	<p>One mark per answer:</p> <p><b>MP1</b> Analysis</p> <p><b>MP2</b> <b>Named</b> document, examples include:</p> <ul style="list-style-type: none"> <li>• A <u>requirement specification</u></li> <li>• Definition of System <b>Objectives</b></li> <li>• List of problems with existing system</li> <li>• Survey results</li> <li>• Feasibility study</li> <li>• Interview notes</li> </ul> <p><b>Max 2 marks</b></p>	<b>2</b>

Question	Answer	Marks
7(b)	<p>One mark for name <b>and</b> use</p> <p>Mark as follows: Examples include:</p> <p>Sub-module: <code>ScanCard()</code> / <code>GetID()</code> Use: Read the barcode from the loyalty card / Get the customer ID from the barcode</p> <p>Sub-module: <code>GetPoints()</code> Use: Get the number of points the customer has</p> <p>Sub-module: <code>ExchangePoints()</code> / <code>UpdateCard()</code> Use: Reduce the number of loyalty points</p> <p>Sub-module: <code>GetDiscount()</code> / <code>CalculateDiscount()</code> Use: Calculate the discount</p> <p>Sub-module: <code>CalculatePoints()</code> Use: Calculate points (following a purchase)</p> <p>Sub-module: <code>UpdatePoints()</code> Use: Update total points a customer has (following a purchase)</p> <p>Sub-module: <code>ShowDiscount()</code> Use: Display the discount</p> <p><b>Max 4 marks</b></p>	<b>4</b>

Question	Answer	Marks
8(a)	<p><b>Example solution</b></p> <pre> PROCEDURE Count(ThisPlayer : INTEGER, ThisRole : STRING)   DECLARE Index, Num, Total : INTEGER    Num ← 0   Total ← 0    FOR Index ← 1 TO 45     IF Character[Index].Player = ThisPlayer AND ____       Character[Index].Role = ThisRole THEN       Num ← Num + 1       Total ← Total + Character[Index].SkillLevel     NEXT Index      IF Num &gt; 0 THEN       OUTPUT "Player ", ThisPlayer, ____         " has ", Num, " characters with the role of         ", ThisRole, " and the total skill level is         ", Total     ELSE       OUTPUT "No characters with that role are         assigned to this player"     ENDIF   ENDPROCEDURE </pre> <p><b>MP1</b> Initialisation of local integers for Num and Total  <b>MP2</b> Loop through 45 elements  <b>MP3</b> Attempt to check Player and Role fields <b>in a loop</b>  <b>MP4</b> Correctly compare Player field with parameter <b>in a loop</b>  <b>MP5</b> Correctly compare Role field with parameter <b>in a loop</b>  <b>MP6</b> ... if player and role found, increment Num <b>and</b> sum Skill Total <b>in a loop</b>  <b>MP7</b> Test for any matches <b>after the loop</b>  <b>MP8</b> <b>Both</b> possible OUTPUT statements correctly formed following an attempt at MP6 but outputting one only</p> <p><b>Max 7 marks</b></p>	<b>7</b>

Question	Answer	Marks
8(b)	<p>Example solution:</p> <pre> PROCEDURE Restore()   DECLARE Index : INTEGER   DECLARE Line  : STRING    OPENFILE "SaveFile.txt" FOR READ   FOR Index ← 1 TO 45     READFILE "SaveFile.txt", Line     Character[Index].Player ← STR_TO_NUM(Extract(Line,   1))      Character[Index].Role ← Extract(Line, 2)     Character[Index].Name ← Extract(Line, 3)     Character[Index].Skill ← STR_TO_NUM(Extract(Line,   4))    NEXT Index    CLOSEFILE "SaveFile.txt"  ENDPROCEDURE </pre> <p>Mark as follows:</p> <p><b>MP1</b> Open the file in read mode and subsequently close</p> <p><b>MP2</b> Loop through 45 elements</p> <p><b>MP3</b> Read a line from the file <b>in a loop</b></p> <p><b>MP4</b> Attempt to use <code>Extract()</code> <b>in a loop</b></p> <p><b>MP5</b> Correct use of <code>Extract()</code> for all fields <b>in a loop</b></p> <p><b>MP6</b> Use of <code>STR_TO_NUM()</code> on Player and Skill <b>in a loop</b></p> <p><b>MP7</b> Completely correct extraction and assignment of all fields <b>in a loop</b></p>	7
8(c)	<p><b>MP1</b> The <u>Character array</u> / <u>character data</u> can be saved before the program is closed</p> <p><b>MP2</b> Allowing the <b>game to continue</b> using the same data / from the point it was saved</p>	2