GradeMax

CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2026**

MARK SCHEME

Maximum Mark: 75

Published

This mark scheme contains the instructions and marking guidance that our examiners used to mark candidate responses for this component.

Before they start marking, examiners complete a standardisation process. They review a range of candidate responses and discuss the mark scheme in detail to agree which answers can and cannot be accepted. Examiners award marks where it is clear that candidate responses have met the requirements of the mark scheme. These requirements may vary between different components or different versions of the same component, depending on the exact requirements of the question and the candidate responses seen during the standardisation process.

Sometimes, our mark schemes may allow marks to be awarded to responses which contain elements that are factually incorrect or for content not covered by the syllabus. We do this in response to the demands of a specific question to support candidates and make sure that our assessments are fair. However, these allowances should not be used as a guide for teaching and learning.

We cannot discuss the mark scheme with schools, and we recommend that schools read the mark scheme together with the assessment material and the Principal Examiner Report for Teachers.

This document consists of **21** printed pages.

© Cambridge University Press & Assessment 2026

[Turn over

GradeMax

General marking principles

All examiners must apply these marking principles when marking candidate responses.

1	Examiners must award marks for each response in line with: • the specific **content** defined in the mark scheme • the specific **skills** defined in the mark scheme or in the level descriptions • the **standard of response** required as exemplified by the standardisation scripts.
2	Examiners must award **whole marks** (not half marks, or other fractions).
3	Examiners must award marks **positively**. This means that: • marks are not deducted for errors or omissions • marks are awarded for correct/valid answers as defined in the mark scheme and also for **other valid answers** which go beyond the scope of the syllabus and mark scheme referring to your team leader as appropriate • the quality of spelling, punctuation and grammar are judged **only** when the mark scheme indicates that these features are **specifically assessed** in the question. However, the meaning of all responses must be unambiguous.
4	Examiners must **consistently** apply the requirements of the mark scheme and any rules for what to do when candidates have not followed instructions correctly.
5	Examiners must use the **full range of marks** defined in the mark scheme, unless limited by the quality of the candidate responses seen.
6	Examiners must award marks according to the mark scheme and **must not** award marks with grade thresholds or grade descriptions in mind.

© Cambridge University Press & Assessment 2026

Annotations guidance for centres

Examiners use a system of annotations as a shorthand for communicating their marking decisions to one another. Examiners are trained during the standardisation process on how and when to use annotations. The purpose of annotations is to inform the standardisation and monitoring processes and guide the supervising examiners when they are checking the work of examiners within their team. The meaning of annotations and how they are used is specific to each component and is understood by all examiners who mark the component.

We publish annotations in our mark schemes to help centres understand the annotations they may see on copies of scripts. Note that there may not be a direct correlation between the number of annotations on a script and the mark awarded. Similarly, the use of an annotation may not be an indication of the quality of the response.

The annotations listed below were available to examiners marking this component in this series.

Annotations

Annotation	Meaning
	Benefit of the doubt
	To indicate where a key word/phrase/code is missing
	Incorrect
	Follow through
	Indicate a point in an answer
Highlighted text	To draw attention to a particular aspect or to indicate where parts of an answer have been combined
	Ignore
	Not answered question
	No examples or not enough
	Not relevant or used to separate parts of an answer
Off-page comment	Allows comments to be entered at the bottom of the RM marking window and then displayed when the associated question item is navigated to.
	Repetition
	Indicates that work or a page has been seen including blank answer spaces and blank pages.
	Correct

© Cambridge University Press & Assessment 2026

GradeMax

Annotation	Meaning
TV	Too vague

PapaCambridge papacambridge.com

GradeMax

Mark scheme abbreviations

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

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

© Cambridge University Press & Assessment 2026

Question	Answer	Marks	Guidance
1(a)(i)	`STRING`	**1**	
1(a)(ii)	`IF` **`LENGTH`**`(Pass) < 6 OR` **`LENGTH`**`(Pass) >` **`8`** `/ >= 9 THEN` Mark as follows: 1 Mark: Use of `LENGTH` function twice 1 Mark: Number **8** used in second condition	**2**	

1(a)(iii)	Mark as follows:
	
	1 `DECLARE LetterFlag, DigitFlag: BOOLEAN`
	1 mark for `BOOLEAN`
	
	2 `LetterFlag ← FALSE`
	1 mark for `FALSE`
	
	3 `FOR Index ← 1 TO LENGTH(Pass)`
	1 mark for `LENGTH`
	
	4 `Character ← MID(Pass, Index, 1)`
	1 mark for `MID`, `Index` and `1`
	
	5 `IF IS_NUM(Character) = TRUE THEN`
	1 mark for `IS_NUM`
	
	6 `DigitFlag ← TRUE`
	`ELSE`
	`LetterFlag ← TRUE`
	1 mark for **both** variables being set to `TRUE`
	
	7 `IF NOT(LetterFlag AND DigitFlag) THEN`
	1 mark for NOT
		7	

1(b)(i)	One mark per state
	
	Example answers:
	1 Sensor activated (mode) / / Intruder/movement detected (mode) // Alert Mode
	2 Alarm sounding // Light flashing (mode) // Alarm activated
	3 Error mode // Power loss detected (mode) // Battery too low mode
	4 Tamper detected (mode) // Lockdown (mode)
	5 Disarm Mode // Maintenance Mode // Password Change Mode // Service Mode
	
	Max 2
		2	

GradeMax

Question	Answer	Marks	Guidance
1(b)(ii)	One mark per event Example answers: 1 Movement / pressure (detected) 2 Window opened / broken (detected) 3 Door open (detected) 4 Entering **wrong** password (repeatedly) 5 Entering password to enter / exit maintenance mode / silence alarm 6 Tampering / damaging alarm (system) 7 Time delay // Time passed after alarm triggered / before alarm activated 8 Battery charge too low 9 Pressing panic button	**3**	Do not accept repeating examples in stem. Reject Turning system on. Reject reference to creating password.

© Cambridge University Press & Assessment 2026

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>1(c)</td><td>Example Solution

<table>
<tr><th>rule being tested</th><th>example of test data</th><th>reason for invalid test data</th></tr>
<tr><td>rule 1</td><td>1A</td><td>password is too short</td></tr>
<tr><td>rule 1</td><td>PHRASE345678</td><td>Password too long</td></tr>
<tr><td>rule 2</td><td>ABCDEF</td><td>Does not contain a digit</td></tr>
<tr><td>rule 2</td><td>1234567</td><td>Does not contain a letter</td></tr>
<tr><td>rule 3</td><td>PASSWRD2</td><td>letters are repeated</td></tr>
<tr><td>rule 3</td><td>ABCDEF11</td><td>Digits / Numbers are repeated</td></tr>
<tr><td>rule 3</td><td>AAA111</td><td>Both digits and letters are repeated

Accept Characters repeated</td></tr>
</table>

Mark as follows:

Rule 1 test:
1 mark for one example of test data for rule 1 and appropriate reason

<table>
<tr><td>Test data</td><td>A string of more than 8 characters that contains both letters and digits.

The letters must be upper case

Each character only occurs once</td></tr>
<tr><td>Reason</td><td>Password is too long</td></tr>
</table>
</td><td>3</td><td>Do not accept once if lower-case letters or characters that are not letters or digits used.

Reject if test data breaks Rule 2 or/and Rule 3</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th><th>Guidance</th></tr>
<tr><td>1(c)</td><td>Rule 2 test (1 mark):
1 mark for two examples of test data that only breaks rule 2 and appropriate reasons.

<table><tr><td>Test data</td><td>A string between 6 and 8 characters long that contains no digits</td></tr><tr><td>Reason</td><td>Does not contain a digit</td></tr><tr><td>Test data</td><td>A string between 6 and 8 characters long that contains no letters</td></tr><tr><td>Reason</td><td>Does not contain a letter</td></tr></table>
Rule 3:
1 mark for two examples of test data for rule 3 and appropriate reasons

<table><tr><td>Test data</td><td>A string between 6 and 8 characters long where only digits are repeated</td></tr><tr><td>Reason</td><td>Digits are repeated</td></tr><tr><td>Test data</td><td>A string between 6 and 8 characters long where both characters and digits are repeated</td></tr><tr><td>Reason</td><td>Both digits and letters are repeated</td></tr></table></td><td></td><td>Reject if letters repeated

Reject if digits repeated

Reject characters repeated

Reject if letters repeated
Reject if no letters</td></tr>
<tr><td>1(d)</td><td>One mark per example

Example answers:
1 Structure chart // Problem / Program broken into modules // Modular Diagram

2 Flowchart
3 Data Flow Diagram
4 Identifier / Variable table // Data structure / Record structure (designs)
5 File structure // Database Design
6 E-R Diagram
7 HCI Design // Report Design
8 Class Diagram // UML Diagram
9 Programming language to use

Max 3</td><td>3</td><td>Reject ‘Decomposition’ by itself
Reject Module Overview by itself</td></tr>
</table>

© Cambridge University Press & Assessment 2026

GradeMax

Question	Answer	Marks	Guidance
1(e)	One mark per stage 1 Analysis 2 Coding // Implementation 3 Maintenance **Max** 2 marks	**2**	NE Development / Programming
1(f)(i)	Integration	**1**	
1(f)(ii)	One mark for the name, one for the description Method: Stub (testing) Description: 1 A simple / dummy module is written to **replace** the module (that is not working) 2 The simple / dummy module will return an expected value / will output a message (to show the module has been called)	**3**	Do not mark description if method incorrect

Question	Answer	Marks	Guidance
2(a)	One mark for either explanation: • To filter out <u>information / data</u> that is **not** necessary for the task • To keep only <u>information / data</u> that **is** necessary for the task **Max 1 mark**	**1**	
2(b)	One mark for each **TWO** data items Example answers 1 Tool ID // Tool name // Type of tool 2 Customer ID // Customer name // Customer address // Account reference 3 Hire ID 4 (Hire) Date 5 Hire time 6 Length of hire: Full or Half day 7 Cost of hire 8 Payment details 9 Tool status **Max 2 marks**	**2**	Reject Quantity Reject Return date if Hire date already given MP 2 Reject just 'Customer'

© Cambridge University Press & Assessment 2026

GradeMax

Question	Answer	Marks	Guidance
2(c)	One mark for each process. Examples include: 1 Input customer details / customer ID / payment details / Tool ID 2 Search for required tool // Checking if tool returned // Checking if tool available // Searching for dates when tool is available 3 Record tool booking // Amend Tool booking // Create tool booking // Record when tool returned // Confirm booking 4 Calculating **time** when tool has to be returned 5 Change / check tool **status** e.g. hired / available / damaged // Prevent same tool being hired twice 6 Process payment / calculate hire cost **Max 2 marks**	**2**	

Question	Answer	Marks	Guidance
3	Example solution Mark points: MP 1 (Step 4) Open `WordList.txt` / File in **read** mode (and subsequently close) MP 2 (Step 5) **Read** the next **line** from the **file** in a loop MP 3 (Step 6) Increment line counter in a loop MP 4 (Step 7) Check if the line read equals the word (input) in a loop MP 5 (Step 8) ...If it does output line counter MP 6 (Step 9) If word not found in file then output `"Not Found"` after all iterations MP 7 (Step 10) Otherwise repeat from step 5 **Max 6 marks from 7 possible marks**	**6**	MP 1 cannot be in a loop structure Do not accept once if MP 2, 3, 4 is **clearly** not in a loop Award only one mark for MP 5 and MP 6 if **both** outputs are possible MP 7 Can be any loop but it **must be clear** where it starts and ends and must include at least one of the end conditions.

© Cambridge University Press & Assessment 2026

Question	Answer	Marks	Guidance
4(a)	After group 1 **back stack** / **forward stack** back stack: www.aaa.home ← TOS; www.ccc.my forward stack: www.bbb.home ← TOS; www.333.corp; www.111.home; www.999.corp; www.222.me Web address currently displayed in the browser: www.zzz.corp After group 2 **back stack** / **forward stack** back stack: www.zzz.corp ← TOS; www.aaa.home; www.ccc.my forward stack: www.333.corp ← TOS; www.111.home; www.999.corp; www.222.me Website displayed in the browser: www.bbb.home After group 3 **back stack** / **forward stack** back stack: www.111.home ← TOS; www.333.corp; www.bbb.home; www.zzz.corp; www.aaa.home; www.ccc.my forward stack: www.222.me ← TOS	**6**	

GradeMax

Question	Answer	Marks	Guidance
4(a)	Website displayed in the browser: www.999.corp Mark as follows: Group 1 1 1 mark for back stack as shown **and** TOS position as shown on both stacks 2 1 mark for forward stack as shown **and** current web address Group 2 3 1 mark for stacks **and** correct TOS pointers as shown **and** current web address Group 3 4 1 mark for back stack as shown **and** TOS as shown 5 1 mark for forward stack as shown **and** TOS as shown 6 1 mark current web address		
4(b)	One mark for identifying an ADT. Queue / Linked List One mark per feature Features of a Queue: 1 1 Each queue element contains one data item 2 A front/start pointer **and** a rear/end pointer 3 Data is added at back / end **and** removed from front / start // works on a FIFO basis 4 Can be **circular or linear** **MAX 2 marks for features** Features of a Linked List: 1 1 Each node / item contains data and a pointer (to the next node/item) 2 A pointer to the start of the list 3 Last node in the list has a null pointer 4 Data may be added / removed by manipulating <u>pointers</u> 5 Pointers are followed to move through nodes in the list (in a given sequence) 6 Unused nodes are stored on a free list // a free-list pointer to the Free List 7 Is a dynamic (data structure) **MAX 2 marks for features**	**3**	Need both Reject use of 2 arrays

Question	Answer	Marks	Guidance

5(a)	The greater than logic operator / `>` in the selection statement would need to be changed to a less than operator / `<`
	
	Or by example
	
	`IF Numbers[Index] > Numbers[Index + 1] THEN`
	
	would be changed to
	
	`IF Numbers[Index] < Numbers[Index + 1] THEN`
	OR
	`IF Numbers[Index] <= Numbers[Index + 1] THEN`
	OR
	`IF Numbers[Index + 1] > Numbers[Index] THEN`
	OR
	`IF Numbers[Index + 1] >= Numbers[Index] THEN`
		1	

5(b)(i)	2 marks for inner loop **MP 1**: The inner loop **iterates** one less time (in algorithm B) each time the outer / repeat loop iterates. **MP 2**: this is because for each **iteration** of the outer / repeat loop (in algorithm B) the last / one more element in the `Numbers` array will be in the correct place. 2 marks for outer loop **MP 3**: The outer / repeat loop (in algorithm B) would stop **iterating** if no swaps are made / all numbers in array are sorted **MP 4**: so not as many **iterations** would be needed (if the `Numbers` array is not initially sorted)	**4**	Reject answers that just repeat the algorithm Reject program stops rather than iteration ends or algorithm ends
5(b)(ii)	**1 mark**: When `Numbers` (array) / array is already in (ascending) order / sorted	**1**	

5(c)(i)	**1 mark:** The parameters / `Num1` and `Num2` are passed by value / `BYVAL`
	// The parameters / `Num1` and `Num2` are **not** passed / should be passed by reference / `BYREF`
	
	1 mark: (The swap made inside) the procedure does **not** change the arguments used /parameters (passed) / the values stored in the `Numbers` elements passed
	// The procedure does **not** swap / change the values stored in the array
		2	

GradeMax

Question	Answer	Marks	Guidance

5(c)(ii)	Solution
	
	`PROCEDURE Swap(BYREF Num1, Num2 : INTEGER)`
	`  DECLARE Temp : INTEGER`
	`  Temp ← Num1`
	`  Num1 ← Num2`
	`  Num2 ← Temp`
	`ENDPROCEDURE`
	
	2 marks: code as above
	
	1 mark: just the procedure header as above
		2	NE for header mark If ByRef is mentioned in the header but syntax is not correct but allow FT for second mark

© Cambridge University Press & Assessment 2026

GradeMax

Question	Answer	Marks	Guidance

6(a)	**Mark Schme One – loop used to form return string**
	(See MS Two for solutions using `LEFT()`/`MID()` to form return string)
	
	Example solution
	
	`FUNCTION AddPadding() RETURNS STRING`
	`  CONSTANT Space = ' '`
	`  DECLARE Plaintext : STRING`
	`  DECLARE LenOfPlaintext : INTEGER`
	`  DECLARE Index : INTEGER`
	`  REPEAT`
	`  OUTPUT "Enter plaintext"`
	`  INPUT Plaintext`
	`  LenOfPlaintext ← LENGTH(Plaintext)`
	`  UNTIL LenOfPlaintext >= 1 AND LenOfPlaintext <= 25`
	`  FOR Index ← LenOfPlaintext + 1 TO 25`
	`  Plaintext ← Plaintext & Space`
	`  NEXT Index`
	`  // Alternative way to append spaces using while loop`
	`  WHILE LENGTH(Plaintext) <> 25`
	`  Plaintext ← Plaintext & Space`
	`  ENDWHILE`
	`  //Alternative way to append spaces`
	`  DECLARE SpaceString : STRING`
	`  SpaceString ← ""`
	`  FOR Index ← LenOfPlaintext + 1 TO 25`
	`  SpaceString ← SpaceString & Space`
	`  NEXT Index`
	`  Plaintext ← Plaintext & SpaceString`
	`  RETURN Plaintext`
	`ENDFUNCTION`
	
	Mark points
	1 Function heading, return type and ending
	2 Prompt for **and** input of plaintext
	3 Conditional loop that checks one of the boundary values
	4 Conditional loop that correctly checks both boundaries of plaintext
	5 …Re-input of plaintext in a loop
	6 Loop that iterates for number of spaces to add to end of plaintext
	7 …Spaces appended to plaintext // Spaces appended to an empty string
	8 Correctly padded plaintext returned following a reasonable attempt at MP7
		8	MP 4 Must be a conditional loop
			If no loop used then MP3, MP4 and MP5 can not be given

© Cambridge University Press & Assessment 2026

Question	Answer	Marks	Guidance

6(a)	**Mark Scheme Two - No Loop used to form return string – use of `LEFT()/MID()` Function**
	
	Example Solution:
	
	`FUNCTION AddPadding() RETURNS STRING`
	`  CONSTANT Fill = "                         " //24 Spaces`
	
	`  DECLARE Plaintext : STRING`
	`  DECLARE LenOfPlaintext : INTEGER`
	`  DECLARE Index : INTEGER`
	
	`  REPEAT`
	`  OUTPUT "Enter plaintext"`
	`  INPUT Plaintext`
	`  LenOfPlaintext ← LENGTH(Plaintext)`
	`  UNTIL LenOfPlaintext >= 1 AND LenOfPlaintext <= 25`
	
	`  Plaintext ← LEFT(Plaintext & Fill, 25)`
	
	`  RETURN Plaintext`
	`ENDFUNCTION`
	
	Mark points
	
	1 Function heading, return type and ending
	2 Prompt for and input of plaintext
	3 Conditional loop that checks one of the boundary values
	Conditional loop that correctly checks for length of plaintext
	4 …Re-input of plaintext **in a loop**
	5 Use of string of at least 24 spaces
	Correct use of `LEFT()` / `MID()` function and `&` to append padding characters
	
	6 Correctly padded plaintext returned following a reasonable attempt at
	MP 6 and MP 7
			Do not accept use of language specific string manipulation which is not the compatible with the pseudocode used for the syllabus

GradeMax

Question	Answer	Marks	Guidance
6(b)(i)	Solutions: • `Grid : ARRAY[1 : 5, 1 : 5] OF CHAR` • `Grid : ARRAY[0 : 4, 0 : 4] OF CHAR` 1 mark for boundaries of both array dimensions: • `[1 : 5, 1 : 5]` • `[0 : 4, 0 : 4]` 1 mark for all other parts of declaration	**2**	Reject use of different brackets eg ( ) Reject `CHARACTER` instead of `CHAR` Reject `STRING` instead of `CHAR` Must be as exactly as above but accept use of lowercase

© Cambridge University Press & Assessment 2026

Question	Answer	Marks	Guidance

6(b)(ii)	**Solution One** : Using nested loop and output of each row in a loop following nested loop
	
	Example Solution
	
	`PROCEDURE PopulateGrid(Plaintext : STRING)`
	`  DECLARE Row, Column, Index : INTEGER`
	`  Index ← 1`
	`  FOR Row ← 1 TO 5`
	`  FOR Column ← 1 TO 5`
	`  Grid[Row, Column] ← MID(Plaintext, Index, 1)`
	`  Index ← Index + 1`
	`  NEXT Column`
	`  Next Row`
	
	`  FOR Row ← 1 TO 5`
	`  OUTPUT Grid[Row, 1], Grid[Row, 2], Grid[Row, 3],_`
	`  Grid[Row, 4], Grid[Row, 5]`
	`  NEXT Row`
	
	`ENDPROCEDURE`
	
	Mark Points
	1 Generate `Index`, 1 to 25, used to extract all characters from `Plaintext`
	
	2 Generate both `Row` and `Col` **in a loop**
	
	3 Extract an individual character from `Plaintext` **in a loop**
	
	4 Attempt to assign the extracted character to `Grid` **in a loop**
	
	5 Extraction **and** assignment of **all** 25 characters in `Plaintext` to `Grid` **in a loop**
	
	6 Attempt to identify / isolate 5 characters comprising at least **one** row
	
	7 Identify / isolate all 5 characters for all 5 rows
	
	8 Output all five rows of `GRID`, each on a separate line
		8	Syntax for `MID()` function must use parameters of correct type – allow other string handling function combinations if they work.
			
			Max 7 if Procedure heading/parameter type incorrect
			Penalise incorrect syntax for indexes i.e. `[Row][Col]` instead once.

Question	Answer	Marks	Guidance

6(b)(ii)	**Solution Two**: Using nested loop and output of each string in a loop following nested loop -
	
	Example Solution
	
	`PROCEDURE PopulateGrid(Plaintext : STRING)`
	
	`  DECLARE Row, Column, Index : INTEGER`
	`  DECLARE RowString : STRING`
	
	`  FOR Row ← 1 TO 5`
	`  FOR Column ← 1 TO 5`
	`  Index ← (Row – 1) * 5 + Column`
	`  Grid[Row, Column] ← MID(Plaintext, Index, 1)`
	`  NEXT Column`
	`  NEXT Row`
	
	`  FOR Row ← 1 TO 5`
	`  RowString ←""`
	`  For Col ← 1 TO 5`
	`  RowString ← RowString & Grid[Row, Column]`
	`  Next Col`
	`  OUTPUT RowString`
	`  NEXT Row`
	
	`ENDPROCEDURE`
	
	Solution Three: Using nested loop where output string is formed in the inner loop and output in string in the outer loop – complex use of Index. No additional loop after nested loop
	
	Example Solution
	
	`PROCEDURE PopulateGrid(Plaintext : STRING)`
	
	`  DECLARE Row, Column, Index : INTEGER`
	`  DECLARE RowString : STRING`
	
	`  FOR Row ← 1 TO 5`
	`  RowString ← ""`
	`  FOR Column ← 1 TO 5`
	`  Index ← (Row – 1) * 5 + Column`
	`  Grid[Row, Column] ← MID(Plaintext, Index, 1)`
	`  RowString ← RowString & MID(Plaintext, Index, 1)`
	`  NEXT Column`
	`  OUTPUT RowString`
	`  NEXT Row`
	`ENDPROCEDURE`		

GradeMax

Question	Answer	Marks	Guidance

6(b)(ii)	**Solution Four: No nested loop**
	
	Example Soution
	
	`PROCEDURE PopulateGrid(Plaintext : STRING)`
	
	`  DECLARE Row, Column, Index : INTEGER`
	
	`  Row ← 1`
	`  Col ← 0`
	`  FOR Index ← 1 TO 25`
	`  IF Index MOD 5 = 0 THEN`
	`  Row ← Row + 1`
	`  Col ← 0`
	`  ENDIF`
	`  Col ← Col + 1`
	`  Grid[Row, Column] ← MID(Plaintext, Index, 1)`
	`  NEXT Index`
	
	`  FOR Row ← 1 TO 5`
	`  OUTPUT Grid[Row, 1], Grid[Row, 2],_`
	`  Grid[Row, 3], Grid[Row, 4], Grid[Row, 5]`
	`  NEXT Row`
	
	`ENDPROCEDURE`		

© Cambridge University Press & Assessment 2026