GradeMax

* 0000800000001 * DFD

CAMBRIDGE
International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

10336233361

COMPUTER SCIENCE **9618/22**

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

2 hours

You must answer on the question paper.

You will need: Insert (enclosed)

INSTRUCTIONS

- Answer **all** questions.
- Use a black or dark blue pen.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen. Do **not** use correction fluid or tape.
- Do **not** write on any bar codes.
- You may use an HB pencil for any diagrams, graphs or rough working.
- Calculators must **not** be used in this paper.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.
- The insert contains all the resources referred to in the questions.

This document has **20** pages. Any blank pages are indicated.

DC (WW) 356722/4
© Cambridge University Press & Assessment 2026

[Turn over

GradeMax

Refer to the **insert** for the list of pseudocode functions and operators.

1 A program is being developed to control a simple alarm system to be used in a house. The alarm has a control panel with a keypad and a screen to display text.

The keypad consists of 36 character keys:

- 26 upper-case letter keys, allowing A to Z to be entered
- 10 digit keys, allowing 0 to 9 to be entered.

(a) When the alarm system is initially installed, a password must be created. When the password is initially entered, it is checked to ensure that it follows a set of validation rules. It is then permanently stored in the alarm system.

For a password to be valid it must follow these validation rules:

Rule 1: contain between six and eight characters inclusive.
Rule 2: contain at least one letter and one digit.
Rule 3: each character can only occur once.

The global variable `Pass` is used to store the password entered.

(i) Complete the pseudocode to declare the variable to be used to hold the password.

```
DECLARE Pass : .....
```

[1]

(ii) Complete the pseudocode to check that a password contains between **six** and **eight** characters inclusive (rule 1).

```
IF ..... (Pass) < 6 OR ..... (Pass) > ..... THEN
   OUTPUT "Error"
   OUTPUT "The password has to contain between 6 and 8 characters."
ENDIF
```

[2]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

(iii) Complete the pseudocode to check that a password entered contains at least one letter and one digit character (rule 2).

```
DECLARE Index : INTEGER

DECLARE Character : CHAR

DECLARE LetterFlag, DigitFlag : .....

DigitFlag ← FALSE

LetterFlag ← .....

FOR Index ← 1 TO ..... (Pass)

   Character ← ..... (Pass, ..... , .....)

   IF ..... (Character) = TRUE THEN

      DigitFlag ← .....

   ELSE

      LetterFlag ← .....

   ENDIF

NEXT Index

IF ..... (LetterFlag AND DigitFlag) THEN

   OUTPUT "Error"

   OUTPUT "Password needs at least one letter and one digit."

ENDIF
```

[7]

(b) A state-transition diagram is being created as part of the design process for the module, which will control the alarm system once it has been successfully installed. Two of the states already identified are that the alarm is in:

- standby mode
- on mode (alarm system is active).

(i) Identify **two** other states that should be part of the state-transition diagram.

1 .....

2 .....

[2]

DO NOT WRITE IN THIS MARGIN

GradeMax

(ii) Two events that will cause a change of state are:

- entering the password when the alarm system is in standby mode
- entering the password when the alarm system is in on mode.

Identify **three** other events that would cause a change of state in the state-transition diagram for the alarm system.

1 .....

2 .....

3 .....

[3]

(c) The validation rules from **1(a)** are given again:

For a password to be valid it must follow these validation rules:

Rule 1: contain between six and eight characters inclusive.
Rule 2: contain at least one letter and one digit.
Rule 3: each character can only occur once.

Complete the table with test data that could be used to test the validation rules.

Each example of test data given must:

- be invalid for the specific rule being tested
- be valid for the other **two** rules **not** being tested
- test a different aspect of the rule being tested.

Two of the tests have been completed for you.

rule being tested	example of test data	reason for invalid test data
rule 1	1A	password is too short
rule 1		
rule 2		
rule 2		
rule 3	PASSWRD2	letters are repeated
rule 3		
rule 3		

[3]

© Cambridge University Press & Assessment 2026

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

GradeMax

(d) Five outcomes that could be produced at the design stage in the program development lifecycle include:

- state-transition diagrams
- pseudocode
- test strategy
- test plan
- structured English.

Identify **three** other outcomes that could be produced at the design stage.

1 .....

2 .....

3 .....

[3]

(e) The design and testing stages are two of the stages in the program development lifecycle used when developing the program to control the alarm system.

Identify **two** other stages in the program development lifecycle.

1 .....

2 .....

[2]

(f) All the individual modules for the alarm system have now been written, tested and debugged as necessary. All these modules are combined to form the program, and further testing is now needed.

(i) Identify this method of testing.

..... [1]

(ii) One of the modules does not work as expected when added to the program.

Identify and describe a testing method that can be used to address this problem, so that the program can continue to be tested.

method .....

description .....

.....

.....

.....

.....

[3]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

GradeMax

2 A new computer system is being developed to help manage a tool hire business. A customer can specify if they want to hire a tool for either half a day or a full day.

An abstract model for the new system is required.

(a) Explain the purpose of abstraction.

.....

..... [1]

(b) State **four** items of data that must be stored each time a tool is hired.

item 1 .....

item 2 .....

item 3 .....

item 4 .....

[2]

(c) The computer system will have to process the tool hire data. One process is to cancel a tool hire booking.

State **two** other processes that would be required.

process 1 .....

.....

process 2 .....

.....

[2]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

GradeMax

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

3 A computer game uses a text file named `WordList.txt` containing 10 000 words, all in lower case. Each word is unique and is stored on a separate line of the text file.

An efficient algorithm is needed to check that a word input by the user is stored in `WordList.txt`. If the word is found, the line number where the word is stored must be output. If the word is **not** found, the message `"Not Found"` must be output.

Complete the steps needed to describe the algorithm.

Do **not** include pseudocode statements in your answer.

The first **three** steps have been completed for you.

step 1: set line counter to 0

step 2: input a word

step 3: convert the word to lower case

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

[6]

© Cambridge University Press & Assessment 2026

4 A desktop web browser stores the history of web addresses visited.

The web browser uses a back button and forward button to move between web pages recently visited.

Two stacks are used when each of these buttons are clicked:

- back stack
- forward stack.

The back button is clicked to return to a previously visited web page.

The back button operates as follows:

- The web address currently being displayed by the browser is pushed onto the forward stack.
- The web address from the top of the back stack is popped.
- The web address popped is loaded by the web browser.

The forward button is clicked to revisit a web page previously visited.

The forward button operates as follows:

- The web address currently being displayed by the browser is pushed onto the back stack.
- The web address from the top of the forward stack is popped.
- The web address popped is loaded by the web browser.

The current state of the two stacks:

back stack		**forward stack**	
		www.333.corp	← TOS
www.zzz.corp	← TOS	www.111.home	
www.aaa.home		www.999.corp	
www.ccc.my		www.222.me	

The top of stack pointer is represented by TOS.

The current web address being displayed by the web browser is: www.bbb.home

The browsing session continues; the buttons are clicked in sequence in the three groups:

group 1:
back button

group 2:
forward button

group 3:
forward button
forward button
forward button

© Cambridge University Press & Assessment 2026

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

GradeMax

(a) Complete the diagrams to show the state of both stacks after each group of operations has been performed.

Note: you must include the top of stack pointer (TOS) and the current web address being displayed for each group diagram.

after group 1

back stack	**forward stack**

Web address currently displayed in the web browser: .....

after group 2

back stack	**forward stack**

Web address currently displayed in the web browser: .....

after group 3

back stack	**forward stack**

Web address currently displayed in the web browser: .....

[6]

© Cambridge University Press & Assessment 2026

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

(b) A stack is an Abstract Data Type (ADT)

Identify **one** other ADT and describe **two** features of the ADT.

ADT .....

feature 1 .....

.....

feature 2 .....

.....

[3]

5 A programmer has defined a global 1D array `Numbers` of type `INTEGER`, which can store 1000 integers. The lower bound of the array `Numbers` is set to 1. Each element of the array has been initialised with a random integer in the range 200 to 99 999.

The programmer has written a bubble sort algorithm to sort the array `Numbers` into ascending order.

Study algorithm A.

Assume all the variables have been correctly declared.

algorithm A:

```
FOR LoopCounter ← 1 TO 999                    //Outer Loop
   FOR Index ← 1 TO 999                       //Inner Loop
      IF Numbers[Index] > Numbers[Index + 1] THEN
         //Start of block to swap neighbouring elements in the array
         Temp ← Numbers[Index]
         Numbers[Index] ← Numbers[Index + 1]
         Numbers[Index + 1] ← Temp
         //End of block to swap neighbouring elements in the array
      ENDIF
   NEXT Index
NEXT LoopCounter
```

(a) Describe the change that would have to be made to the bubble sort algorithm (algorithm A) so that the `Numbers` array would be sorted into descending order.

.....

..... [1]

DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

(b) The programmer rewrites the bubble sort algorithm to make it more efficient.

Study algorithm B.

Assume all the variables have been correctly declared.

algorithm B:

```
Boundary ← 999
REPEAT                                  //Outer Loop
   NoSwaps ← TRUE
   FOR Index ← 1 TO Boundary            //Inner Loop
      IF Numbers[Index] > Numbers[Index + 1] THEN
         //Start of block to swap neighbouring elements in the array
         Temp ← Numbers[Index]
         Numbers[Index] ← Numbers[Index + 1]
         Numbers[Index + 1] ← Temp
         //End of block to swap neighbouring elements in the array
         NoSwaps ← FALSE
      ENDIF
   NEXT Index
   Boundary ← Boundary - 1
UNTIL NoSwaps = TRUE
```

(i) Explain why the second bubble sort algorithm (algorithm B) is more efficient than the first bubble sort algorithm (algorithm A). Your answer must refer to both inner and outer loops.

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

(ii) State the situation that will result in the outer loop iterating for the minimum number of times when the second bubble sort algorithm (algorithm B) is used.

.....

..... [1]

DO NOT WRITE IN THIS MARGIN

(c) The programmer has defined a new procedure:

```
PROCEDURE Swap(Num1 : INTEGER, Num2 : INTEGER)
   DECLARE Temp : INTEGER
   Temp ← Num1
   Num1 ← Num2
   Num2 ← Temp
ENDPROCEDURE
```

The programmer replaces the block of pseudocode in the algorithm that swaps the values in two neighbouring elements in the array with a call to this procedure.

The block of code:

```
Temp ← Numbers[Index]
Numbers[Index] ← Numbers[Index + 1]
Numbers[Index + 1] ← Temp
```

Is replaced by:

```
CALL Swap(Numbers[Index], Numbers[Index + 1])
```

The `Swap()` procedure will **not** work as expected when used in the bubble sort algorithm.

(i) Explain why the procedure `Swap()` will **not** work as expected.

.....

.....

.....

..... [2]

(ii) Write pseudocode to amend the version of the `Swap()` procedure given so that it will work as expected.

The amended version of the `Swap()` procedure must **not** use global variables.

.....

.....

.....

.....

.....

.....

..... [2]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

GradeMax

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

Turn over

6 A program is being developed to encrypt a message.

The following terms are used in the encryption technique to be used:

term	definition
plain text	The original message before it is encrypted.
cipher text	The encrypted version of the plain text.

The encryption method to be used is based on the algorithm for the Caesar Box Cipher. All plain text messages to be encrypted must be 25 characters in length or less. If a message is less than 25 characters in length, then spaces are added to the end to make the padded plain text the required length.

A square grid of five columns and five rows is used. The padded plain text message is put into the grid row by row. The cipher text is obtained by reading the grid column by column.

For example:

plain text: `This is a message`
The number of characters in the original plain text, including spaces, is 17.

plain text with added spaces at the end: `This␣is␣a␣message␣␣␣␣␣␣␣␣`
The character ␣ is used to indicate a space in the padded plain text.

The grid filled, row by row, with the padded plain text.

T	h	i	s	␣
i	s	␣	a	␣
m	e	s	s	a
g	e	␣	␣	␣
␣	␣	␣	␣	␣

The cipher text is obtained by reading the grid column by column.
cipher text: `Timg␣hsee␣i␣s␣␣sas␣␣␣␣a␣␣`

The programmer has defined the first program module as:

module	description
`AddPadding()`	• user prompt for and input of plain text • check plain text contains between 1 and 25 characters inclusive • re-input of plain text until it contains between 1 and 25 characters • if required, additional space characters are added to the end of the plain text to ensure it contains exactly 25 characters • return of amended plain text

© Cambridge University Press & Assessment 2026

GradeMax
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

GradeMax

(a) Write pseudocode for module `AddPadding()`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

(b) The programmer has defined a global 2D array `Grid` to be used to create the cipher text.

(i) Complete the pseudocode statement used to declare the global 2D array `Grid`

`DECLARE` .....

.....

[2]

(ii) The programmer has defined an additional program module as follows:

module	description

`PopulateGrid()`	• called with one parameter of type string representing the correctly padded plain text
	• populate the array `Grid` using the plain text so that it can be used at a later point to create the cipher text
	• output the contents of the array `Grid` so it is displayed row-by-row
	
	using the original example for the padded plain text:
	`This∆is∆a∆message∆∆∆∆∆∆∆`
	
	the output would be:
	`This`
	`is a`
	`messa`
	`ge`

Write pseudocode for module `PopulateGrid()`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

© Cambridge University Press & Assessment 2026

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

GradeMax

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

PapaCambridge
papacambridge.com

GradeMax

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

GradeMax

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

© Cambridge University Press & Assessment 2026

GradeMax

BLANK PAGE

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (Cambridge University Press & Assessment) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in our Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge International Education is the name of our awarding body and a part of Cambridge University Press & Assessment, which is a department of the University of Cambridge.

© Cambridge University Press & Assessment 2026