GradeMax

* 0000800000001 * DFD

Cambridge Assessment International Education

Cambridge International AS & A Level

CANDIDATE NAME

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

7630789161

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2025**

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 or correction fluid.
- 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 (CJ) 343237/2
© UCLES 2025

[Turn over

GradeMax

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

1 (a) The table contains pseudocode examples.

Each example may contain statements that relate to one or more of:

- selection
- iteration (repetition)
- subroutines (procedures or functions).

Complete the table by placing **one or more** ticks ('✓') in each row.

Pseudocode example	Selection	Iteration	Subroutine

`IF Status = FALSE THEN`
`  FOR Count ← 1 TO 20`
`  CALL Reset(Count)`
`  NEXT Count`
`ENDIF`			

`OTHERWISE : Status ← TRUE`			
`WHILE AllDone() = TRUE`			
`30 : NextChar ← 'X'`			

[4]

(b) Complete the table by giving the appropriate data type.

Variable	Example data value	Data type
`Result`	`5.42`	
`MonthLetter`	`"JFMAMJJASOND"`	
`Birthday`	`15/11/2009`	

[3]

(c) Evaluate each expression in the table by using the data values shown in **(b)**.

Write 'ERROR' if the expression contains an error.

Expression	Evaluates to
`INT(Result) + 1 > 6`	
`NUM_TO_STR(LENGTH(MonthLetter))`	
`NUM_TO_STR(Result + "3.2")`	
`MID(MonthLetter, MONTH(Birthday) – 2, 1)`	

[4]

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

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

2 `Data` is a global 1D array containing 20 elements of type `REAL`

An algorithm will:

- input a sequence of real values, one at a time
- assign each value to consecutive array elements, starting from index 1
- end when the value 99.9 is input, or all 20 elements have been assigned (the value 99.9 must **not** be stored in the array).

Complete the program flowchart to represent the algorithm:



[5]

GradeMax

3 A text file `OldFile.txt` contains IDs, names and email addresses for members of a club. Three information items are stored for each member and each item is stored on a separate line.

The example shows the information for the first two members in the first six lines of the file:

Line in file	Information item	Example data
1	member 1 ID	`"AB1234"`
2	member 1 name	`"Freddie Jones"`
3	member 1 email address	`"FreddieJ909@Cambridge.org"`
4	member 2 ID	`"BC2345"`
5	member 2 name	`"Sue Smith"`
6	member 2 email address	`"Sue1024@Cambridge.org"`

The member ID string is always two letters followed by four digits.

The file design is to be changed so that information for each user is stored as a single line of the file, with the character `'\'` used as a separator between data items.

For example, the single line for member 1 will be:

```
"AB1234\Freddie Jones\FreddieJ909@Cambridge.org"
```

An algorithm will produce a new file `NewFile.txt` from the contents of `OldFile.txt`

Assume:

- `LineX`, `LineY` and `LineZ` are of type `STRING` and are used to store the three items of information for each member
- `NewString` is a temporary variable of type `STRING`
- `OldFile.txt` exists and contains valid data.

(a) The algorithm to create the new file is expressed in steps.

Complete the following numbered steps:

Step 1 : open the file ..... in .....

Step 2 : open the file ..... in .....

Step 3 : read a line from ..... and store in .....

Step 4 : read a line from ..... and store in .....

Step 5 : read a line from ..... and store in .....

Step 6 : set `NewString` to .....

Step 7 : write `NewString` to .....

Step 8 : repeat from step ..... until .....

Step 9 : close both files.

[6]

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

(b) The character `'\'` has been chosen as a separator.

Explain why this is a suitable character.

.....

..... [1]

(c) Two new information items are to be stored for each member. Both new items are encrypted; they can each contain any character (including `'\'`) and can be of any length up to a maximum of 99 characters.

For example:

Information item	Example data
member 1 ID	`"AB1234"`
member 1 name	`"Freddie Jones"`
member 1 email address	`"FreddieJ909@Cambridge.org"`
member 1 new information 1	`"En/98&*(|?\/D7iP"`
member 1 new information 2	`"23\CoboL"`

Explain the additional file design changes needed so that:

- all the information items for each member are stored on a single line of `NewFile.txt`
- it is possible to extract each information item after the line is read.

Assume the `'\'` character is not used in any email address.

.....

.....

.....

.....

.....

.....

.....

.....

..... [3]

GradeMax

4 A quiz has nine questions. There are:

- five easy questions each worth 3 points
- four hard questions each worth 5 points.

At the end of the quiz the points for each correctly answered question are added to give a total score.

A check is made to test that the total score is valid using a 1D array `CheckTotal` of type `BOOLEAN`. Each index value of the array represents a possible total score. The corresponding element value is `TRUE` if the index value represents a valid total score and `FALSE` otherwise.

The first nine rows of the array are:

Index value	Element value	Comment
0	`TRUE`	valid total score (no correct answers)
1	`FALSE`	invalid total score
2	`FALSE`	invalid total score
3	`TRUE`	valid total score (one 3-point question correct)
4	`FALSE`	invalid total score
5	`TRUE`	valid total score (one 5-point question correct)
6	`TRUE`	valid total score (two 3-point questions correct)
7	`FALSE`	invalid total score
8	`TRUE`	valid total score (one 3-point question and one 5-point question correct)

For example, a total score of 6 points is valid; the value of the array element at index value 6 is `TRUE`

(a) Write pseudocode to declare `CheckTotal` and to set all elements of the array to `FALSE`

All variables used must be declared.

.....

.....

.....

.....

.....

.....

.....

..... [4]

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 pseudocode represents an algorithm to set the appropriate elements of the array to `TRUE`

Complete the pseudocode:

```
DECLARE EasyQ, HardQ : INTEGER

FOR EasyQ ← ..... TO 15 STEP .....

   FOR HardQ ← 0 TO ..... STEP .....

      CheckTotal[.....] ← TRUE

   NEXT HardQ

NEXT EasyQ
```

[5]

(c) The array elements have been assigned the required values.

A module `ValidateScore()` will take an integer value representing a total score as a parameter. It will return `TRUE` if the total score is valid, or `FALSE` if it is **not** valid.

Describe the algorithm for `ValidateScore()` using **four** steps.

Do **not** use pseudocode in your answer.

Step 1 .....

.....

Step 2 .....

.....

Step 3 .....

.....

Step 4 .....

.....

[4]

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

5 A program is developed to satisfy a specific customer requirement.

The project follows a program development life cycle model. This model divides the development process into several different stages.

(a) The table lists some of the development activities.

Complete the table by writing the name of the life cycle stage for each activity:

Activity	Name of life cycle stage
a structure chart is produced	
a program is modified to allow it to run on new hardware	
the programmer identifies the customer's requirements	
an Integrated Development Environment (IDE) provides context-sensitive help such as 'auto-complete'	

[4]

(b) Alpha and beta testing has been completed. The final testing stage is carried out by the customer.

Identify this final testing stage.

..... [1]

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

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

GradeMax

6 A string function `Compare()` will compare two strings.

The function will:

- take **four** parameters:
  - two strings, `String1` and `String2`
  - a character, `Position`, to indicate whether `String1` will be compared with the start (`'s'`), or the end (`'e'`) of `String2`
  - a Boolean, `CaseMatters`, to indicate whether an upper case character and lower case character (for example, `'A'` and `'a'`) are regarded as different (`TRUE`), or as the same (`FALSE`)
- return `FALSE` if `String2` has fewer characters than `String1`
- return `TRUE` if the comparison is true, otherwise return `FALSE`

For example:

Parameter				Return value
String1	**String2**	**CaseMatters**	**Position**	
"Cat"	"Catalogue"	TRUE	's'	TRUE
"CAT"	"Catalogue"	TRUE	's'	FALSE
"CAT"	"Catalogue"	FALSE	's'	TRUE
"Cat"	"Catalogue"	TRUE	'e'	FALSE
"GUE"	"Catalogue"	FALSE	'e'	TRUE
"Catalogue"	"Cat"	TRUE	's'	FALSE

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

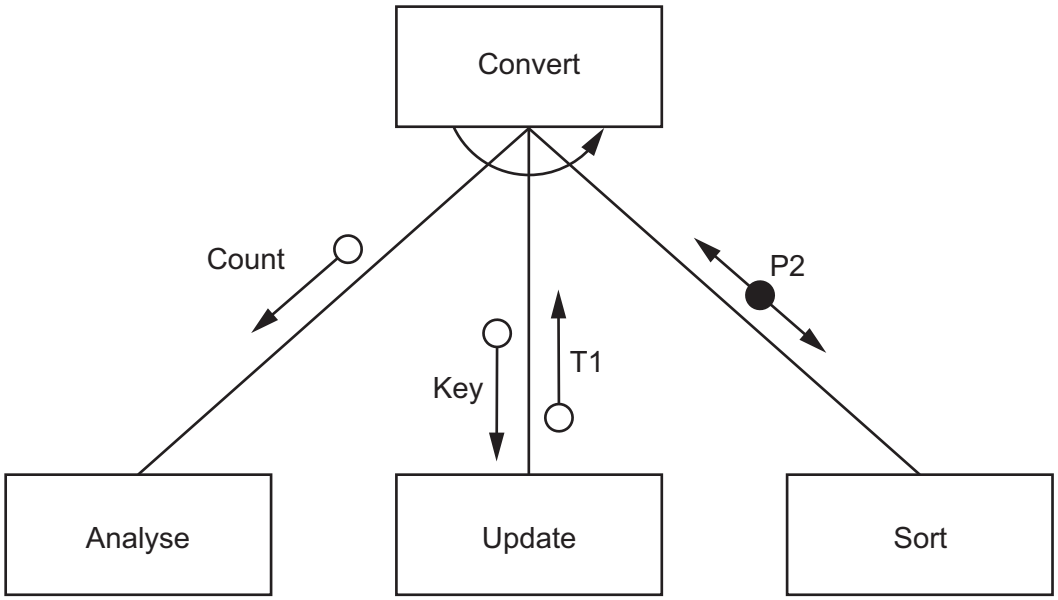
Write pseudocode for the function `Compare()`

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

..... [7]

GradeMax

7 The structure chart shows part of a program design:



(a) Explain the meaning of the curved arrow symbol in this structure chart.

.....

.....

.....

..... [2]

(b) The program designer has noted that:

- Count is the number of elements in an array
- T1 is a value representing the number of elements in an array that have been modified
- Key is of type `STRING`

Write the pseudocode module headers for analyse, update and sort.

Analyse:

.....

.....

Update:

.....

.....

Sort:

.....

.....

[4]

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

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

GradeMax

8 A program is being developed to manage student book loans from a college library.

The programmer has defined a record type to define each loan.

The data items are:

Data item	Data type	Comment
`StudentID`	`STRING`	the unique ID of the student who has borrowed the book
`BookID`	`STRING`	the unique ID of the book being borrowed
`OnLoan`	`BOOLEAN`	`TRUE` if the book has **not** been returned

The programmer has defined a global array `Loan` to store 7000 loan records.

There are more elements in the array than books in the library. Unused elements have the `StudentID` set to an empty string. These may occur anywhere in the array.

The programmer has defined a program module:

Module	Description
`CountLoans()`	• called with a parameter of type `STRING` representing a `StudentID` • counts the number of books currently on loan to the specified student • counts the number of books that the student has already returned • output both counts together with a suitable message

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 `CountLoans()`

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

[7]

GradeMax

(b) As a reminder, a global array `Loan` stores 7000 loan records with data items for each loan record:

Data item	Data type	Comment
`StudentID`	`STRING`	the unique ID of the student who has borrowed the book
`BookID`	`STRING`	the unique ID of the book being borrowed
`OnLoan`	`BOOLEAN`	`TRUE` if the book has **not** been returned

A new module is defined:

Module	Description
`NewLoan()`	• called with two parameters of type `STRING` representing a `StudentID` and a `BookID` • searches the array for an unused loan record • if found, updates the loan record and returns `TRUE`, otherwise returns `FALSE`

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

Write efficient pseudocode for `NewLoan()`

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [7]

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

(c) When a book is returned, the loan record will have its `OnLoan` data set to `FALSE`

A new procedure `Archive()` will mark these records as unused after first saving them for future reference.

Explain how these records can be saved for future reference.

.....

.....

.....

..... [2]

(d) It is decided to extend the program so that a new module `Reminder()` will send an email to the student three days before the book is due to be returned.

Outline the changes that will need to be made to the data stored and how this data will be used to generate the reminder email.

Data .....

.....

.....

.....

Use .....

.....

.....

.....

[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

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

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 (UCLES) 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 the Cambridge Assessment International Education 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 Assessment International Education is part of Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

© UCLES 2025