GradeMax

CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **May/June 2026**

2 hours 30 minutes

You will need: Candidate source files (listed on page 2)
evidence.docx

INSTRUCTIONS

- Carry out every instruction in each task.
- Save your work using the file names given in the task as and when instructed.
- You must **not** have access to either the internet or any email system during this examination.
- You must save your work in the evidence document as stated in the tasks. If work is **not** saved in the evidence document, you will **not** receive marks for that task.
- You must use a high-level programming language from this list:
  - Java (console mode)
  - Python (console mode)
  - Visual Basic (console mode)
- A mark of **zero** will be awarded if a programming language other than those listed here is used.

INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].

This document has **12** pages. Any blank pages are indicated.

DC (EV) 356378/3
© Cambridge University Press & Assessment 2026

[Turn over

You have been supplied with the following source file:

`Toys.txt`

Open the evidence document, **evidence.docx**

Make sure that your name, centre number and candidate number will appear on every page of this document. This document must contain your answers to each question.

Save this evidence document in your work area as:
evidence_ followed by your centre number_candidate number, for example: **evidence_zz999_9999**

A class declaration can be used to declare a record. If the programming language used does **not** support arrays, a list can be used instead.

One source file is used to answer **Question 2**. The file is called **`Toys.txt`**

1 A program stores integer data in a linear queue.

The queue is created as a global 1D array with space for 10 elements.

The queue has a global head pointer, initialised to `-1`, that points to the first element in the queue.

The queue has a global tail pointer, initialised to `-1`, that points to the last element in the queue.

(a) The main program initialises the 10 elements of the queue to `-1` and initialises the two pointers.

Write program code to declare the global variables and initialise the queue and its pointers.

Save your program as **Question1_J26**. Copy and paste the program code into **1(a)** in the evidence document.

[3]

(b) The function `Enqueue()` performs these tasks:

- takes an integer parameter
- checks if the queue is full
- returns Boolean `FALSE`, if the queue is full
- stores the parameter in the queue, updates the appropriate pointer(s) and returns Boolean `TRUE`, if the queue is **not** full.

Write program code for `Enqueue()`

Save your program. Copy and paste the program code into **1(b)** in the evidence document.

[7]

© Cambridge University Press & Assessment 2026

GradeMax

(c) (i) Write program code to extend the main program to:

- take 11 integers as input from the user
- use the appropriate function to insert each integer into the queue
- output a message if at any point the queue is full
- output the final content of the queue array. Do **not** dequeue the data.

You do **not** need to validate the data input.

Save your program.

Copy and paste the program code into **1(c)(i)** in the evidence document.

[5]

(ii) Test your program.

Enter the following data as input in the order given:

1

2

3

4

5

6

7

8

9

10

11

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot into **1(c)(ii)** in the evidence document.

[1]

© Cambridge University Press & Assessment 2026

(d) The function `Dequeue()`:

- returns `-1` if the queue is empty
- updates the pointer(s) and returns the next item from the queue if the queue is **not** empty.

Write program code for `Dequeue()`

Save your program. Copy and paste the program code into **1(d)** in the evidence document.

[6]

(e) **(i)** Write program code to amend the main program to:

- take 6 integers as input from the user and store them in the queue using the appropriate function
- use the appropriate function to remove 4 elements from the queue
- output each removed element from the queue.

Save your program. Copy and paste the program code into **1(e)(i)** in the evidence document.

[3]

(ii) Test your program.

Enter the following data as input in the order given:

100

10

5

6

18

68

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into **1(e)(ii)** in the evidence document.

[1]

© Cambridge University Press & Assessment 2026

GradeMax

2 A program stores data about toys that are made from bricks. The toys are stored in a record structure with four fields:

- `ToyName`: the string name of the toy
- `NumberBricks`: the integer number of bricks in the toy
- `Difficulty`: the string difficulty level of the toy (easy, medium or expert)
- `Category`: the string category of the toy (for example town, display, play).

(a) Write program code to declare a record structure with the identifier `ToyRecord` and the **four** fields.

Save your program as **Question2_J26**. Copy and paste the program code into **2(a)** in the evidence document.

[2]

(b) The text file `Toys.txt` stores data about **16** toys. Each toy is on a new line in the file. The data is stored in the order: toy name, number of bricks, difficulty, category. Each value is separated by a comma.

For example, the first line in the file is:

```
Fire Station,3981,Expert,Town
```

The name of the toy is Fire Station. The toy has 3981 bricks. The difficulty level is Expert. The category is Town.

(i) The function `ReadData()`:

- takes an array of type `ToyRecord` as a parameter
- opens the file `Toys.txt` and reads in the data
- creates a new record of type `ToyRecord` for each toy read from the file
- stores each record in the parameter array
- returns the populated array.

The function uses exception handling when opening and reading from the file.

Write program code for `ReadData()`

Save your program. Copy and paste the program code into **2(b)(i)** in the evidence document.

[8]

© Cambridge University Press & Assessment 2026

GradeMax

(ii) The procedure `OutputToys()` takes an array of type `ToyRecord` as a parameter and outputs the data for each toy in the array.

Write program code for `OutputToys()`

Save your program. Copy and paste the program code into **2(b)(ii)** in the evidence document.

[3]

(iii) Write program code for the main program to:

- declare an array of type `ToyRecord` with space for **16** elements
- call the function `ReadData()` with the empty array and store the returned array
- call `OutputToys()` with the returned array.

Save your program. Copy and paste the program code into **2(b)(iii)** in the evidence document.

[3]

(iv) Test your program.

Take a screenshot of the output(s).

Save your program. Copy and paste the screenshot into **2(b)(iv)** in the evidence document.

[1]

(c) The program needs to allow a user to enter the difficulty level of a toy and then write all toys that match that difficulty level to an external text file.

(i) The function `SearchToys()`**:**

- takes an array of type `ToyRecord` as a parameter
- prompts the user to enter the difficulty level
- takes the difficulty level as input until it is valid
- creates a new array containing the `ToyRecord` objects that match the difficulty level input
- returns the populated array.

Write program code for `SearchToys()`

Save your program. Copy and paste the program code into **2(c)(i)** in the evidence document.

[5]

© Cambridge University Press & Assessment 2026

(ii) The procedure `WriteToys()`:

- takes an array of type `ToyRecord` as a parameter
- creates and opens the file `SelectedToys.txt` to write
- writes each record from `ToyRecord` to `SelectedToys.txt` with each field separated by a comma and each record on a new line. For example:

```
<ToyName>,<NumberBricks>,<Difficulty>,<Category>
```

Write program code for `WriteToys()`

Save your program. Copy and paste the program code into **2(c)(ii)** in the evidence document.

[3]

(iii) Write program code to extend the main program to:

- call `SearchToys()` with the populated array and store the return array
- call `WriteToys()` with the returned array.

Save your program. Copy and paste the program code into **2(c)(iii)** in the evidence document.

[2]

(iv) Test your program. Enter the appropriate input for a medium difficulty level when prompted.

Take **one** screenshot of the text file the program produces with the contents and filename clearly visible.

Save your program. Copy and paste the screenshot into **2(c)(iv)** in the evidence document.

[1]

© Cambridge University Press & Assessment 2026 **[Turn over**

3 A prototype for a game is designed using Object-Oriented Programming (OOP).

The game will allow a character to move along a path (for example a road or sidewalk).

The prototype requires the classes setting up and the path creating ready for the character to be implemented.

(a) Each space in the path can contain an object. The path and objects are created as classes.

The class `TheObject` stores the data about the objects. The attributes are public.

TheObject	
`ObjectType : String`	stores the type of object, either `"Money"` or `"Experience"`
`Amount : Integer`	stores the amount of the object, for example `200`
`Constructor()`	initialises the attributes to the parameter values

The class `Path` stores the data about the objects. The attributes are public.

Path	
`BoardArray[0:99] : TheObject`	an array of 100 objects of type `TheObject`
`NumberSpaces : Integer`	stores the number of spaces in the path
`Constructor()`	initialises `NumberSpaces` to the parameter value. Initialises all 100 array objects to an empty object of type `TheObject`. An empty object has the values `ObjectType = ""` and `Amount = -1`
`GetNumberSpaces()`	returns `NumberSpaces`
`InitialiseData()`	takes an array of objects of type `TheObject` as a parameter and stores them in `BoardArray`
`ReturnPath()`	returns a string created using data from each object in `BoardArray`
`GetPathObject()`	takes a number as a parameter and returns the object in that index in `BoardArray`

(i) Write program code to declare the class `TheObject` and its constructor.

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_J26**.

Copy and paste the program code into **3(a)(i)** in the evidence document.

[4]

© Cambridge University Press & Assessment 2026

(ii) Write program code to declare the class `Path`, its constructor and method `GetNumberSpaces()`

Use your programming language appropriate constructor.

Do **not** declare the other methods.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into **3(a)(ii)** in the evidence document.

[4]

(iii) The method `InitialiseData()` takes an array of type `TheObject` as a parameter.

The method stores each element from the parameter array into the array `BoardArray` in the same order.

If there are more objects in the parameter array than in `NumberSpaces`, the method does **not** store the data and returns `FALSE`. Otherwise, it returns `TRUE`

Write program code for `InitialiseData()`

Save your program.

Copy and paste the program code into **3(a)(iii)** in the evidence document.

[3]

(iv) The method `ReturnPath()` creates and returns a string containing the type and amount of each object that is **not** a null object in `BoardArray`, in the format:

```
<ObjectType>-<Amount>
```

For example, an array with two **not** null objects produces the string:

```
Money-100 Experience-50
```

You can assume that there will always be an object that is **not** null in `BoardArray`

Write program code for `ReturnPath()`

Save your program.

Copy and paste the program code into **3(a)(iv)** in the evidence document.

[2]

© Cambridge University Press & Assessment 2026 **[Turn over**

(v) The method `GetPathObject()` takes an integer index as a parameter and returns the object at the parameter index in `BoardArray`

Write program code for `GetPathObject()`

Save your program.

Copy and paste the program code into **3(a)(v)** in the evidence document.

[1]

(b) (i) Write program code for the main program to:

- create a new array, `ArrayOfObjects`, of type `TheObject` with **five** spaces
- store the following objects in each space in `ArrayOfObjects` in the order given:
  money with amount 50
  experience with amount 100
  money with amount 200
  money with amount 20
  experience with amount 30
- create a new object, `Path1`, of type `Path` with **five** spaces
- call `InitialiseData()` for `Path1` with `ArrayOfObjects`
- call `ReturnPath()` for `Path1` and output the return value.

Save your program.

Copy and paste the program code into **3(b)(i)** in the evidence document.

[6]

(ii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot into **3(b)(ii)** in the evidence document.

[1]

© Cambridge University Press & Assessment 2026

BLANK PAGE

© Cambridge University Press & Assessment 2026

GradeMax

BLANK PAGE

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