Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/43**

Paper 4 Practical **May/June 2023**

2 hours 30 minutes

1947047972

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

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.

Open the evidence document **evidence.doc**

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

One source file is used to answer **Question 1** and two source files are used to answer **Question 3**. The files are called **`Data.txt`**, **`AnimalData.txt`** and **`ColourData.txt`**

A class declaration can be used to declare a record.

A list is an alternative to an array.

1 A program reads data from a file and searches for specific data.

(a) The main program needs to read 25 integer data items from the text file `Data.txt` into a local 1D array, `DataArray`

(i) Write program code to declare the local array `DataArray`

Save your program as **Question1_J2023**. Copy and paste the program code into **part 1(a)(i)** in the evidence document.

[1]

(ii) Amend the main program to read the contents of `Data.txt` into `DataArray`

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

[4]

(b) **(i)** The procedure `PrintArray()` takes an integer array as a parameter and outputs the contents of the array in the order they are stored.

The items are printed on the same line, for example:

10 4 5 13 25

Write program code for the procedure `PrintArray()`

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

[3]

(ii) Amend the main program to output the contents of `DataArray` using the procedure `PrintArray()`

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

[1]

(iii) Test your program.

Take a screenshot of the output.

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

[1]

(c) The function `LinearSearch()`:

- takes an integer array and integer search value as parameters
- counts and returns the number of times the search value is found in the array.

Write program code for the function `LinearSearch()`

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

[3]

(d) (i) Amend the main program to:

- prompt the user to input a whole number between 0 and 100 inclusive
- read and validate the input from the user
- call `LinearSearch()` with `DataArray` and the validated input value
- output the result in the format:
  `The number 7 is found 2 times.`

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

[4]

(ii) Test your program by inputting the number 12.

Take a screenshot of the output.

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

[1]

2 A computer game is being designed that will include different vehicles. A prototype for the game is being developed using object-oriented programming.

The class `Vehicle` stores data about the vehicles. Each vehicle has an identification name, a maximum speed, a current speed and a horizontal position. The value `IncreaseAmount` is added to the current speed each time the vehicle increases its speed.

Vehicle	
`ID : STRING`	stores the identification name for the vehicle
`MaxSpeed : INTEGER`	stores the maximum speed
`CurrentSpeed : INTEGER`	stores the current speed
`IncreaseAmount : INTEGER`	stores the amount `CurrentSpeed` increases by
`HorizontalPosition : INTEGER`	stores the horizontal position
`Constructor()`	initialises `ID`, `MaxSpeed` and `IncreaseAmount` to the parameter values initialises both `CurrentSpeed` and `HorizontalPosition` to `0`
`GetCurrentSpeed()`	returns the current speed
`GetIncreaseAmount()`	returns the increase amount
`GetHorizontalPosition()`	returns the horizontal position
`GetMaxSpeed()`	returns the maximum speed
`SetCurrentSpeed()`	assigns the parameter to the current speed
`SetHorizontalPosition()`	assigns the parameter to the horizontal position
`IncreaseSpeed()`	calculates and stores the new speed and horizontal position of the vehicle

(a) (i) Write program code to declare the class `Vehicle`. All attributes must be private.

You only need to declare the class and its constructor. Do not declare any other methods.

Use your programming language's appropriate constructor.

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

Save your program as **Question2_J2023**.

Copy and paste the program code into **part 2(a)(i)** in the evidence document.

[5]

(ii) Write program code for the get methods `GetCurrentSpeed()`, `GetIncreaseAmount()`, `GetMaxSpeed()` and `GetHorizontalPosition()`

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

[3]

(iii) Write program code for the set methods `SetCurrentSpeed()` and `SetHorizontalPosition()`

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

[3]

(iv) The method `IncreaseSpeed()`:

- adds `IncreaseAmount` to the current speed
- adds the updated current speed to the horizontal position.

The current speed of a vehicle cannot exceed its maximum speed.

Write program code for the method `IncreaseSpeed()`

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

[3]

(b) The child class `Helicopter` inherits from the parent class `Vehicle`. A helicopter also has a vertical position and changes the vertical position when it increases speed.

Helicopter	
`VerticalPosition : INTEGER`	stores the vertical position
`VerticalChange : INTEGER`	stores the amount `VerticalPosition` changes by
`MaxHeight : INTEGER`	stores the maximum height the helicopter can reach
`Constructor()`	takes the ID, maximum speed, increase amount, vertical change and maximum height as parameters initialises the vertical position to 0
`GetVerticalPosition()`	returns the vertical position
`IncreaseSpeed()`	changes the current speed, horizontal and vertical position of the helicopter

(i) Write program code to declare the class `Helicopter`. You only need to declare the class and its constructor. You do not need to declare the other methods.

Use your programming language's appropriate constructor.

All attributes must be private.

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

Save your program.

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

[5]

(ii) The `Helicopter` method `IncreaseSpeed()` overrides the method from the parent class and:

- adds the amount of vertical change to the vertical position
- adds `IncreaseAmount` to the current speed
- adds the updated current speed to the horizontal position.

The vertical position of a helicopter cannot exceed its maximum height.

The current speed of a helicopter cannot exceed its maximum speed.

Write program code for the method `IncreaseSpeed()`

Save your program.

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

[4]

(c) A procedure needs to output the horizontal position and speed of a vehicle. If the vehicle is a helicopter, it also outputs the vertical position.

All outputs must include appropriate messages.

Write program code for this procedure.

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

[3]

(d) The main program needs to:

- instantiate a car as a new vehicle with the ID "Tiger", a maximum speed of 100 and an increase amount of 20
- instantiate a new helicopter with the ID "Lion", a maximum speed of 350, an increase amount of 40, a vertical change of 3 and a maximum height of 100
- call `IncreaseSpeed()` twice for the car and then call the output procedure from **part 2(c)** for the car
- call `IncreaseSpeed()` twice for the helicopter and then call the output procedure from **part 2(c)** for the helicopter.

(i) Write program code for the main program.

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

[5]

(ii) Test your program.

Take a screenshot of the output.

Save your program. Copy and paste the screenshot into **part 2(d)(ii)** in the evidence document.

[1]

3 A program implements two stacks using 1D arrays. One stack stores the names of colours. One stack stores the names of animals.

(a) The program contains the following global arrays and variables:

- 1D array `Animal` to store the names of up to 20 animals.
- 1D array `Colour` to store the names of up to 10 colours.
- `AnimalTopPointer` to point to the next free space in the array `Animal`, initialised to 0.
- `ColourTopPointer` to point to the next free space in the array `Colour`, initialised to 0.

Write program code to declare the global arrays and variables.

Save your program as **Question3_J2023**.

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

[3]

(b) (i) Study the pseudocode function `PushAnimal()`:

```
FUNCTION PushAnimal(DataToPush : STRING) RETURNS BOOLEAN
   IF AnimalTopPointer = 20 THEN
      RETURN FALSE
   ELSE
      Animal[AnimalTopPointer] ← DataToPush
      AnimalTopPointer ← AnimalTopPointer + 1
      RETURN TRUE
   ENDIF
ENDFUNCTION
```

Write program code for the function `PushAnimal()`

Save your program.

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

[3]

(ii) Study the pseudocode function `PopAnimal()`:

```
FUNCTION PopAnimal() RETURNS STRING
   DECLARE ReturnData : STRING
   IF AnimalTopPointer = 0 THEN
      RETURN ""
   ELSE
      ReturnData ← Animal[AnimalTopPointer - 1]
      AnimalTopPointer ← AnimalTopPointer - 1
      RETURN ReturnData
   ENDIF
ENDFUNCTION
```

Write program code to declare the function `PopAnimal()`

Save your program. Copy and paste the program code into **part 3(b)(ii)** in the evidence document.

[3]

(iii) The procedure `ReadData()`:

- reads the animal names from the file `AnimalData.txt`
- uses `PushAnimal()` to insert each name onto the stack
- uses appropriate exception handling if the file does not exist.

Write program code for the procedure `ReadData()`

Save your program. Copy and paste the program code into **part 3(b)(iii)** in the evidence document.

[5]

(iv) The function `PushColour()` performs the same actions as `PushAnimal()` but inserts an item into `Colour`.

The function `PopColour()` performs the same actions as `PopAnimal()` but removes the next item from `Colour`.

Write program code for the functions `PushColour()` and `PopColour()`

Save your program. Copy and paste the program code into **part 3(b)(iv)** in the evidence document.

[2]

(v) Amend the procedure `ReadData()` so that it also:

- reads the colours from the text file `ColourData.txt`
- uses `PushColour()` to insert each colour onto the stack
- uses appropriate exception handling if the file does not exist.

Save your program.

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

[2]

(c) The procedure `OutputItem()`:

- pops the next item from both `Animal` and `Colour`
- outputs the colour and animal on one line, for example `"black horse"`

If there is no data in `Colour`:

- the animal is pushed back onto `Animal`
- `"No colour"` is output.

If there is no data in `Animal`:

- the colour is pushed back onto `Colour`
- `"No animal"` is output.

Write program code for the procedure `OutputItem()`

Save your program.

Copy and paste the program code into **part 3(c)** in the evidence document.

[5]

(d) The main program:

- calls the procedure `ReadData()`
- calls `OutputItem()` **four** times.

(i) Write program code for the main program.

Save your program.

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

[1]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

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

[1]

BLANK PAGE

BLANK PAGE