Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/43**

Paper 4 Practical **May/June 2022**

2 hours 30 minutes

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 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

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.

A source file is used to answer question **1**. The file is called **`HighScore.txt`**

1 The text file `HighScore.txt` stores the players who have scored the top ten scores in a game, in descending order of score. The file stores the 3-character name of the player, and their integer score, in the order: player, score.

For example, the current top player in the text file:

`FYI` is the player name

`10000` is the score

The program:

- reads in the data from `HighScore.txt`
- allows the user to enter a new player name and their score
- if appropriate, inserts the new player (name and score) into the top ten
- writes the top ten players (name and score) into a new text file `NewHighScore.txt`

(a) The program stores the players and their scores in an array of 11 elements (10 elements to be read from the file, 1 element to be inserted by the user).

Write a program to declare one or more arrays, as global data structures, to store the player names and their scores.

Save your program as **Question1_J2022**. Copy and paste the program code into **part 1(a)** in the evidence document.

[2]

(b) The procedure `ReadHighScores()` opens the file `HighScore.txt` and reads the data into the data structure(s) declared in **part 1(a)**.

Write program code to declare the procedure `ReadHighScores()`.

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

[6]

(c) The procedure `OutputHighScores()` outputs all the values in the data structure(s) in the format:

```
PlayerName Score
```

For example, the first two data items:

```
FYI 10000
ABC 9092
```

Write program code to declare the procedure `OutputHighScores()`.

Save your program.

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

[3]

(d) The main program should first call `ReadHighScores()` and then `OutputHighScores()`.

(i) Write the program code for the main program.

Save your program.

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

[2]

(ii) Test your program.

Take a screenshot to show the output from **part 1(d)(i)**.

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

[1]

(e) The main program needs to ask the user to input a new player name and a score. If this score is in the top ten then it will create a new top ten list that includes this score.

(i) Amend the main program to ask the user to input a 3-character player name and an integer score that must be between 1 and 100000 inclusive.

Save your program.

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

[3]

(ii) Write program code to declare a procedure that:

- takes the player name and score as parameters
- creates a new top ten list that includes the parameter if appropriate.

Save your program.

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

[5]

(iii) Amend the main program to call the procedure from **part 1(e)(ii)**.

Output the contents of the array before inserting the new player name and score, and output the contents of the array after inserting the new player name and score.

Save your program.

Copy and paste the program code into **part 1(e)(iii)** in the evidence document.

[2]

(iv) Test your program by entering the player name `"JKL"` and the score `"9999"`.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 1(e)(iv)** in the evidence document.

[1]

(f) The procedure `WriteTopTen()` stores the new top ten player names and scores in a text file called `NewHighScore.txt`

Write program code to declare the procedure `WriteTopTen()`.

Save your program.

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

[4]

2 A computer game is being developed using object-oriented programming.

One element of the game is a balloon. This is designed as the class `Balloon`.

The class has the following attributes and methods.

Balloon	
`Health : INTEGER`	The health of the balloon
`Colour : STRING`	The colour of the balloon
`DefenceItem : STRING`	The item the balloon uses to defend itself
`Constructor()`	Initialises the defence item and colour using the parameters Initialises health to 100
`ChangeHealth()`	Takes the change as a parameter and adds this to the health
`GetDefenceItem()`	Returns the defence item of the object
`CheckHealth()`	If the health is 0 or less, it returns `TRUE`, otherwise it returns `FALSE`

(a) The constructor takes the name of the defence item and the balloon's colour as parameters and sets these to the attributes. The health is initialised to `100`.

Write program code to declare the class `Balloon` and its constructor. Do not write any other methods.

Use your language appropriate constructor.

All attributes should be private. If you are writing in Python include attribute declarations using comments.

Save your program as **Question2_J2022**.

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

[5]

(b) The get method `GetDefenceItem()` returns the defence item of the object.

Amend your program code to include the get method `GetDefenceItem()`.

Save your program.

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

[2]

(c) The object's method `ChangeHealth()` takes an integer number as a parameter and adds this to the health attribute of the object.

Amend your program code to include the method `ChangeHealth()`.

Save your program.

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

[2]

(d) The object's method `CheckHealth()` returns `TRUE` if the health of the object is 0 or less (no health remaining) and returns `FALSE` otherwise (health remaining).

Amend your program code to include the method `CheckHealth()`.

Save your program.

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

[2]

(e) Amend the main program to:

- take as input a defence item and colour from the user
- create a new balloon with the identifier `Balloon1` using the data input.

Save your program.

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

[3]

(f) The function `Defend()`:

- takes a balloon object as a parameter
- takes as input the strength of an opponent from the user
- uses the `ChangeHealth()` method to subtract the strength input from the object's health
- outputs the defence item of the balloon
- checks the health of the object and outputs an appropriate message if it has no health remaining, or if it has health remaining
- returns the amended balloon object.

Write program code to declare the function `Defend()`.

Save your program.

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

[8]

(g) (i) Amend the main program to call the function `Defend()`.

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

[2]

(ii) Test your program using the following inputs:

- balloon defence method `"Shield"`
- balloon colour `"Red"`
- strength of opponent `50`

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(g)(ii)** in the evidence document.

[1]

3 A program uses a circular queue to store strings. The queue is created as a 1D array, `QueueArray`, with 10 string items.

The following data is stored about the queue:

- the head pointer initialised to 0
- the tail pointer initialised to 0
- the number of items in the queue initialised to 0.

(a) Declare the array, head pointer, tail pointer and number of items.

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

Save your program as **Question3_J2022**.

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

[2]

(b) The function `Enqueue` is written in pseudocode. The function adds `DataToAdd` to the queue. It returns `FALSE` if the queue is full and returns `TRUE` if the item is added.

The function is incomplete, there are **five** incomplete statements.

```
FUNCTION Enqueue(BYREF QueueArray[] : STRING, BYREF HeadPointer : INTEGER,
                 BYREF TailPointer : INTEGER, NumberItems : INTEGER,
                 DataToAdd : STRING) RETURNS BOOLEAN

  IF NumberItems = ..... THEN

    RETURN .....

  ENDIF

  QueueArray[.....] ⟵ DataToAdd

  IF TailPointer >= 9 THEN

    TailPointer ⟵ .....

  ELSE

    TailPointer ⟵ TailPointer + 1

  ENDIF

  NumberItems ⟵ NumberItems .....

  RETURN TRUE

ENDFUNCTION
```

Write program code for the function `Enqueue()`.

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

[7]

(c) The function `Dequeue()` returns `"FALSE"` if the queue is empty, or it returns the next data item in the queue.

Write program code for the function `Dequeue()`.

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

[6]

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

- take as input 11 string values from the user
- use the `Enqueue()` function to add each element to the queue
- output an appropriate message to state whether each addition was successful, or not
- call `Dequeue()` function twice and output the return value each time.

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

[5]

(ii) Test your program with the input data:

```
"A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K"
```

Take a screenshot to show the output.

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

[1]

BLANK PAGE

BLANK PAGE