Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **October/November 2022**

2 hours 30 minutes

8012187262

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**

A class declaration can be used to declare a record.

If the programming language does not support arrays, a list can be used instead.

A source file is used to answer **Question 2**. The file is called **`Characters.txt`**

1 A computer program is needed to store jobs in order of priority. Each job has a job number (for example, 123) and a priority from 1 to 10, with 1 being the highest priority and 10 the lowest.

The program stores the jobs in a global 2D array.

The pseudocode declaration for the array is:

```
DECLARE Jobs : ARRAY[0:99, 0:1] OF INTEGER
```

For example:

- `Jobs[0, 0]` stores the job number of the first job.
- `Jobs[0, 1]` stores the priority of the first job.

The global variable, `NumberOfJobs`, stores the number of jobs currently in the array.

(a) Write program code to declare the global 2D array `Jobs` and the global variable `NumberOfJobs`.

Save your program as **Question1_N22**. Copy and paste the program code into **part 1(a)** in the evidence document.

[3]

(b) The procedure `Initialise()` stores `-1` in each of the array elements and assigns `0` to `NumberOfJobs`.

Write program code for the procedure `Initialise()`.

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

[3]

(c) The procedure `AddJob()`:

- takes a job number and priority as parameters
- stores the job in the next free array element
- outputs 'Added' if the job was successfully stored in the array
- outputs 'Not added' if the job was not successfully stored in the array.

Write program code for the procedure `AddJob()`.

Save your program.

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

[5]

(d) The main program should call the procedure `Initialise()` and then use the `AddJob()` procedure to store the following jobs in the order given:

Job number	Priority
12	10
526	9
33	8
12	9
78	1

Write program code for the main program and perform the tasks described.

Save your program.

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

[2]

(e) When a new job has been added, the array is sorted into ascending numerical order of priority using an insertion sort.

Write program code for the procedure `InsertionSort()` to sort the data into ascending numerical order of priority.

Save your program.

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

[5]

(f) The procedure `PrintArray()` outputs each job number and priority on a line, for example:

```
123 priority 1

39 priority 3

120 priority 7
```

Write program code for the procedure `PrintArray()`.

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

[3]

(g) The main program needs to sort the array and then output the contents of the array.

(i) Amend the main program by writing program code to call procedures `InsertionSort()` and `PrintArray()`.

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

[1]

(ii) Test your program.

Take a screenshot of the output.

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

[1]

2 A computer game is being developed. The game has 10 different characters that are all active in the game.

Part of the game is being written using object-oriented programming.

The class `Character` stores data about the characters. Each character has a name and the x coordinate and y coordinate of their current position.

Character	
`Name : STRING`	stores the name of the character
`XCoordinate : INTEGER`	stores the x coordinate
`YCoordinate : INTEGER`	stores the y coordinate
`Constructor()`	initialises `Name`, `XCoordinate` and `YCoordinate` from the values passed as parameters
`GetName()`	returns the name of the character
`GetX()`	returns the x coordinate of the character
`GetY()`	returns the y coordinate of the character
`ChangePosition()`	takes `XChange` as an integer parameter and adds it to the x coordinate takes `YChange` as an integer parameter and adds it to the y coordinate

(a) Write program code to declare the class `Character` and its constructor. Do **not** write program code for the other methods.

Use your programming language appropriate constructor.

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

Save your program as **Question2_N22**.

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

[4]

(b) Write program code for the **three** get methods for the class `Character`.

Save your program.

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

[3]

(c) Write program code for the method `ChangePosition()`.

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

[2]

(d) The main program has a 1D array of characters. Each character is stored as an object of type `Character`.

The game has a maximum of 10 characters. The character names, x coordinates and y coordinates are stored in the file `Characters.txt` in the order:

- name
- x coordinate
- y coordinate.

For example, the first character in the file is named Amal, with the x coordinate 0 and the y coordinate 2.

Amend the main program by writing program code to:

- declare the array
- read in all 10 characters from `Characters.txt`
- store each character as an object in the array.

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

[7]

(e) The main program needs to read in a character's name from the user, search for the character in the array and store the index of its position. It repeats until the user enters a name that exists in the array.

Amend the main program by writing program code to perform this task.

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

[5]

(f) The user will enter a letter to identify the direction the chosen character from **part 2(e)** should move.

- If 'A' is input, the character moves left (x coordinate minus 1).
- If 'W' is input, the character moves up (y coordinate plus 1).
- If 'S' is input, the character moves down (y coordinate minus 1).
- If 'D' is input, the character moves right (x coordinate plus 1).

Amend the main program by writing program code to:

- take a letter as input until it is a valid move (A, W, S or D)
- change the position of the character using the appropriate method.

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

[7]

(g) **(i)** When a change to a character's position has been made, the program needs to output the character's name and the new x and y coordinates of the character, in the format:

```
Qui has changed coordinates to X = 83 and Y = 0
```

Amend the main program by writing program code to perform these tasks.

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

[2]

(ii) Test your program by inputting the following **four** items of data in the order given:

```
THOMAS
qui
X
A
```

Take a screenshot of the output.

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

[1]

3 A program uses a linear queue to store up to 100 integers.

(a) A 1D array, `Queue`, is used to store the data. The head pointer points to the first number stored in the queue and the tail pointer points to the next free space in the queue.

Write program code to:

- declare the global array `Queue`
- declare the global variable head pointer and assign an appropriate initial value
- declare the global variable tail pointer and assign an appropriate initial value.

Save your program as **Question3_N22**. Copy and paste the program code into **part 3(a)** in the evidence document.

[3]

(b) The function `Enqueue()` takes an integer value as a parameter and stores it in the queue. It returns `TRUE` if the value was successfully stored and `FALSE` otherwise.

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

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

[6]

(c) The main program uses the `Enqueue()` function to store the numbers 1 to 20 (inclusive) in the queue, in ascending numerical order. The program should output 'Successful' if all numbers are successfully enqueued, and 'Unsuccessful' otherwise.

Amend the main program by writing program code to perform this task.

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

[4]

(d) The following iterative pseudocode function calculates the total of all the values stored in the queue.

```
FUNCTION IterativeOutput(Start: INTEGER) RETURNS INTEGER

   DECLARE Total : INTEGER

   Total ← 0

   FOR Count ← Start - 1 TO HeadPointer STEP -1

      Total ← Total + Queue[Count]

   NEXT Count

   RETURN Total

ENDFUNCTION
```

Rewrite the function as a recursive function using program code.

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

[6]

(e) The main program calls the recursive function from **part 3(d)** and outputs the value returned.

(i) Amend the main program by writing program code to perform this task.

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

[1]

(ii) Test your program.

Take a screenshot to show the output.

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

[1]

BLANK PAGE

BLANK PAGE

BLANK PAGE