Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **October/November 2021**

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 **2(e)**. The file is called `Pictures.txt`

1 Study the following pseudocode for a recursive function.

```
FUNCTION Unknown(BYVAL X, BYVAL Y : INTEGER) RETURNS INTEGER
  IF X < Y THEN
    OUTPUT X + Y
    RETURN (Unknown(X + 1, Y) * 2)
  ELSE
    IF X = Y THEN
      RETURN 1
    ELSE
      OUTPUT X + Y
      RETURN (Unknown(X - 1, Y) DIV 2)
    ENDIF
  ENDIF
ENDFUNCTION
```

The operator `DIV` returns the integer value after division e.g. `13 DIV 2` would give 6

(a) Write program code to declare the function `Unknown()`.

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

[3]

(b) The main program needs to run all **three** of the following function calls and output the result of each call:

```
Unknown(10, 15)
Unknown(10, 10)
Unknown(15, 10)
```

(i) For each of the **three** function calls, the main program needs to:

- output the value of the two parameters
- call the function with those parameters
- output the return value.

Write the program code for the main program.

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

[3]

(ii) Take a screenshot to show the output from **part (b)(i)**.

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

[2]

(c) Rewrite the function `Unknown()` as an iterative function, `IterativeUnknown()`.

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

[7]

(d) The iterative function needs to be called **three** times with the same parameters as in **part (b)**.

(i) For each of the **three** function calls, the main program needs to:

- output the value of the two parameters
- call the iterative function with those parameters
- output the return value.

Amend the main program to perform these tasks.

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

[1]

(ii) Take one or more screenshots to show the output of both functions for each set of parameters.

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

[1]

2 A program, written using object-oriented programming, stores pictures as objects.

The program stores the dimensions of the picture (width and height), the colour of the frame (e.g. black), and a description of the picture (e.g. flowers).

The class has the following attributes and methods.

Picture	

`Description : STRING`
`Width : INTEGER`
`Height : INTEGER`
`FrameColour : STRING`
	`// stores a description of the picture`
	`// stores the width e.g. 30`
	`// stores the height e.g. 40`
	`// stores the colour e.g. black`

`Constructor()`
 	
`GetDescription()`
`GetHeight()`
`GetWidth()`
`GetColour()`
`SetDescription()`
	`// takes all four values as parameters and sets them to the private attributes`
	`// returns the description of the picture`
	`// returns the height`
	`// returns the width`
	`// returns the frame colour`
	`// takes the new description as a parameter and writes the value to description`

(a) The constructor takes the picture description, frame colour, height, and width as parameters and sets these to the private attributes.

Write the program code to declare the class `Picture` 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 programming language, include attribute declarations using comments.

Save your program as **question 2**.

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

[5]

(b) The four get methods return the associated attribute, for example, `GetDescription()` returns the description of the picture.

Write the **four** get methods.

Save your program.

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

[3]

(c) The method `SetDescription()` takes a new description as a parameter, and writes this value to the appropriate attribute.

Write the method `SetDescription()`.

Save your program.

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

[2]

(d) Write program code to declare an array of type `Picture` with 100 elements.

Save your program.

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

[1]

(e) The text file `Pictures.txt` stores the data for the pictures in the order: description, width, height, colour.

For example, for the first picture in the text file:

`Flowers` is the description
`45` is the width
`50` is the height
`black` is the frame colour.

The data read into the program from the text file is stored in an array of type `Picture`.
The main program and the function will need to access the array data.

The function `ReadData()`:

- opens the file `Pictures.txt`
- reads the data from the file
- creates a new object of type `Picture` for each picture
- writes each object to the array
- raises an exception if the file cannot be found
- counts and returns the number of pictures in the array.

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

Save your program.

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

[8]

(f) The main program calls the function `ReadData()`.

Write the main program.

Save your program.

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

[2]

(g) The main program needs to ask the user to input their requirements for a picture. The user will enter the colour of the frame, the maximum width, and the maximum height of the picture.

The program will then search the array of pictures, and output the picture description, the width, and the height of any picture that meets the user's requirements.

The program should allow the user to input the colour in any case (e.g. Silver, silver, or SILVER), and still output the correct results.

Edit the main program to perform the described actions.

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

[7]

(h) Test your program by inputting the following search criteria:

- BLACK, 100, 100
- silver, 25, 25

Take screenshots to show the output for both search criteria.

Copy and paste the screenshots into **part 2(h)** in the evidence document.

[2]

BLANK PAGE

3 An ordered binary tree stores integer data in ascending numerical order.

The data for the binary tree is stored in a 2D array with the following structure:

	LeftPointer	Data	RightPointer
Index	**[0]**	**[1]**	**[2]**
[0]	1	10	2
[1]	-1	5	-1
[2]	-1	16	-1

Each row in the table represents one node on the tree.
The number `-1` represents a null pointer.

(a) The 2D array, `ArrayNodes`, is declared with space for 20 nodes.

Each node has a left pointer, data and right pointer.

The program also initialises the:

- `RootPointer` to `-1` (null); this points to the first node in the binary tree
- `FreeNode` to `0`; this points to the first empty node in the array.

Write program code to declare `ArrayNodes`, `RootPointer` and `FreeNode` in the main program.

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

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

[4]

(b) The procedure `AddNode()` adds a new node to the array `ArrayNodes`.

The procedure needs to:

- take the array, root pointer and free node pointer as parameters
- ask the user to enter the data and read this in
- add the node to the root pointer if the tree is empty
- otherwise, follow the pointers to find the position for the data item to be added
- store the data in the location and update all pointers.

There are **six** incomplete statements in the following pseudocode for the procedure AddNode().

```
PROCEDURE AddNode(BYREF ArrayNodes[] : ARRAY OF INTEGER,
                  BYREF RootPointer : INTEGER, BYREF FreeNode : INTEGER)
  OUTPUT "Enter the data"
  INPUT NodeData
  IF FreeNode <= 19 THEN
    ArrayNodes[FreeNode, 0] ⟵ -1
    ArrayNodes[FreeNode, 1] ⟵ ……
    ArrayNodes[FreeNode, 2] ⟵ -1
    IF RootPointer = …… THEN
      RootPointer ⟵ 0
    ELSE
      Placed ⟵ FALSE
      CurrentNode ⟵ RootPointer
      WHILE Placed = FALSE
        IF NodeData < ArrayNodes[CurrentNode, 1] THEN
          IF ArrayNodes[CurrentNode, 0] = -1 THEN
            ArrayNodes[CurrentNode, 0] ⟵ ……
            Placed ⟵ TRUE
          ELSE
            …… ⟵ ArrayNodes[CurrentNode, 0]
          ENDIF
        ELSE
          IF ArrayNodes[CurrentNode, 2] = -1 THEN
            ArrayNodes[CurrentNode, 2] ⟵ FreeNode
            Placed ⟵ ……
          ELSE
            CurrentNode ⟵ ArrayNodes[CurrentNode, 2]
          ENDIF
        ENDIF
      ENDWHILE
    ENDIF
    FreeNode ⟵ …… + 1
  ELSE
    OUTPUT "Tree is full"
  ENDIF
ENDPROCEDURE
```

Write **program code** for the procedure AddNode().

Save your program.

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

[8]

(c) The procedure `PrintAll()` outputs the data in each element in `ArrayNodes`, in the order they are stored in the array.

Each element is printed in a row in the order:

```
LeftPointer  Data  RightPointer
```

For example:

```
 1        20       -1
-1        10       -1
```

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

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

[4]

(d) The main program should loop 10 times, each time calling the procedure `AddNode()`. It should then call the procedure `PrintAll()`.

(i) Edit the main program to perform the actions described.

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

[3]

(ii) Test the program by entering the data:

10
5
15
8
12
6
20
11
9
4

Take a screenshot to show the output after the given data are entered.

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

[1]

(e) An in-order tree traversal visits the left node, then the root (and outputs this), then visits the right node.

(i) Write a recursive procedure, `InOrder()`, to perform an in-order traversal on the tree held in `ArrayNodes`.

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

[7]

(ii) Test the procedure `InOrder()` with the same data entered in **part (d)(ii)**.

Take a screenshot to show the output after entering the data.

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

[1]

BLANK PAGE