Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **October/November 2023**

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 **16** pages. Any blank pages are indicated.

Open the evidence document, **evidence.doc**

Make sure that your name, centre number and candidate number 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 1.** The file is called **`StackData.txt`**

1 A program stores lower-case letters in two stacks.

One stack stores vowels (a, e, i, o, u) and one stack stores consonants (letters that are not vowels).

Each stack is implemented as a 1D array.

(a) (i) Write program code to declare two 1D global arrays: `StackVowel` and `StackConsonant`.

Each array needs to store up to 100 letters. The index of the first element in each array is 0.

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

Save your program as **Question1_N23**. Copy and paste the program code into part **1(a)(i)** in the evidence document.

[2]

(ii) The global variable `VowelTop` is a pointer that stores the index of the next free space in `StackVowel`.

The global variable `ConsonantTop` is a pointer that stores the index of the next free space in `StackConsonant`.

`VowelTop` and `ConsonantTop` are both initialised to 0.

Write program code to declare and initialise the two variables.

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

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

[1]

(b) (i) The procedure `PushData()` takes one letter as a parameter.

If the parameter is a vowel, it is pushed onto `StackVowel` and the relevant pointer updated.

If the stack is full, a suitable message is output.

If the parameter is a consonant, it is pushed onto `StackConsonant` and the relevant pointer updated.

If the stack is full, a suitable message is output.

You do **not** need to validate that the parameter is a letter.

Write program code for `PushData()`.

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

[6]

(ii) The file `StackData.txt` stores 100 lower-case letters.

The procedure `ReadData()` reads each letter from the file and uses `PushData()` to push each letter onto its appropriate stack.

Use appropriate exception handling if the file does not exist.

Write program code for `ReadData()`.

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

[6]

(c) The function `PopVowel()` removes and returns the data at the top of `StackVowel` and updates the relevant pointer(s).

The function `PopConsonant()` removes and returns the data from the top of `StackConsonant` and updates the relevant pointer(s).

If either stack is empty, the string `"No data"` must be returned.

Write program code to declare `PopVowel()` and `PopConsonant()`.

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

[5]

(d) The program first needs to call `ReadData()` and then:

1. prompt the user to input their choice of vowel or consonant
2. take, as input, the user's choice
3. depending on the user's choice, call `PopVowel()` or `PopConsonant()` and store the return value.

The three steps are repeated until 5 letters have been successfully returned and stored.

If either stack is empty at any stage, an appropriate message must be output.

Once 5 letters have been successfully returned and stored, they are output on one line, for example:

```
abyti
```

(i) Write program code for the main program.

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

[6]

(ii) Test your program with the following inputs:

```
vowel

consonant

consonant

vowel

vowel
```

Take a screenshot of the output.

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

[1]

BLANK PAGE

2 An integer is said to be divisible by another integer if the result of the division is also an integer.

For example:

10 is divisible by 1, 2, 5 and 10:

- 10 ÷ 1 = 10
- 10 ÷ 2 = 5
- 10 ÷ 5 = 2
- 10 ÷ 10 = 1

10 is not divisible by 4:

- 10 ÷ 4 = 2.5

1, 2, 5 and 10 are said to be the divisors of 10.

The iterative function `IterativeCalculate()` totals all the divisors of its integer parameter and returns this total.

Example 1: if the parameter is 10, the total will be 18 (1 + 2 + 5 + 10) .
Example 2: if the parameter is 4, the total will be 7 (1 + 2 + 4) .

A pseudocode algorithm for `IterativeCalculate()` is shown.

```
FUNCTION IterativeCalculate(Number : INTEGER) RETURNS INTEGER

   DECLARE Total : Integer

   DECLARE ToFind : Integer

   ToFind ⟵ Number

   Total ⟵ 0

   WHILE Number <> 0

      IF ToFind MODULUS Number = 0 THEN

         Total ⟵ Total + Number

      ENDIF

      Number ⟵ Number - 1

   ENDWHILE

   RETURN Total

ENDFUNCTION
```

The operator `MODULUS` calculates the remainder when one number is divided by another.

(a) (i) Write program code for `IterativeCalculate()`.

Save your program as **Question2_N23**. Copy and paste the program code into part **2(a)(i)** in the evidence document.

[5]

(ii) Write program code to call `IterativeCalculate()` with 10 as the parameter and output the return value.

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

[2]

(iii) Test your program.

Take a screenshot of the output.

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

[1]

(b) `IterativeCalculate()` has been rewritten as the recursive function `RecursiveValue()`.

A pseudocode algorithm for `RecursiveValue()` is given. The function is incomplete.

```
FUNCTION RecursiveValue(Number : INTEGER, ToFind : INTEGER)
         RETURNS INTEGER

   IF Number = .............. THEN

      RETURN 0

   ELSE

      IF ToFind .............. Number = 0 THEN

         RETURN .............. + RecursiveValue(Number - 1, ToFind)

      ELSE

         .............. RecursiveValue(Number - 1, ..............)

      ENDIF

   ENDIF

ENDFUNCTION
```

(i) Write program code for `RecursiveValue()`.

Save your program.

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

[7]

(ii) Write program code to call `RecursiveValue()` with 50 as the first parameter and 50 as the second parameter and output the return value.

Save your program.

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

[1]

BLANK PAGE

3 A computer game is written using object-oriented programming.

The game has multiple characters.

The class `Character` stores data about the game characters. Each character has a name, date of birth, intelligence value and speed value.

Character	
`CharacterName : STRING`	stores the name of the character
`DateOfBirth : DATE`	stores the date of birth of the character
`Intelligence : REAL`	stores the intelligence value of the character
`Speed : INTEGER`	stores the speed value of the character
`Constructor()`	initialises `CharacterName`, `DateOfBirth`, `Intelligence` and `Speed` to the parameter values
`SetIntelligence()`	assigns the value of the parameter to `Intelligence`
`GetIntelligence()`	returns the value of `Intelligence`
`GetName()`	returns the name of the character
`ReturnAge()`	calculates and returns the age of the character as an integer
`Learn()`	increases the value of `Intelligence` by 10%

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

Do **not** declare the other methods.

Use your programming language's appropriate constructor.

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

Save your program as **Question3_N23**.

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

[5]

(ii) The get methods `GetIntelligence()` and `GetName()` return the attribute values.

Write program code for the methods `GetIntelligence()` and `GetName()`.

Save your program.

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

[3]

(iii) The method `SetIntelligence()` assigns the value of its parameter to the attribute.

Write program code for `SetIntelligence()`.

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

[2]

(iv) The method `Learn()` increases the current value of `Intelligence` by 10%.

Write program code for `Learn()`.

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

[1]

(v) The method `ReturnAge()` calculates and returns the age of the character in years as an integer.

Assume that the current year is 2023 and **only** use the year from the date of birth for the calculation. For example, the method returns 18 if the character was born on any date in 2005.

Write program code for the method `ReturnAge()`.

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

[2]

(b) (i) Write program code to create a new instance of `Character` with the identifier `FirstCharacter`.

The name of the character is Royal, date of birth is 1 January 2019, intelligence is 70 and speed is 30.

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

[2]

(ii) Write program code to call the method `Learn()` for the character created in part **3(b)(i)**.

Output the name, age and intelligence of the character in an appropriate message.

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

[3]

(iii) Test your program.

Take a screenshot of the output.

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

[1]

(c) The class `MagicCharacter` inherits from the class `Character`. A magic character has an element, for example, water. This element changes how they learn. The magic character's element is stored in the additional attribute `Element`.

MagicCharacter	
`Element : STRING`	stores the element for the character
`Constructor()`	takes `Element`, `CharacterName`, `DateOfBirth`, `Intelligence` and `Speed` as parameters calls its parent class constructor with the appropriate values initialises `Element` to its parameter value
`Learn()`	alters the intelligence of the character depending on the character's element

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

Do **not** declare the other method.

Use your programming language's appropriate constructor.

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

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

[5]

(ii) The method `Learn()` overrides the parent class method and increases the intelligence depending on the character's element.

- If the element is fire or water, intelligence increases by 20%.
- If the element is earth, intelligence increases by 30%.
- If the element is not fire, water or earth the intelligence increases by 10%.

Write program code for `Learn()`.

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

[3]

(d) (i) Write program code to create a new instance of `MagicCharacter` with the identifier `FirstMagic`.

The name of the character is Light, date of birth is 3 March 2018, intelligence is 75, speed is 22 and element is fire.

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

[2]

(ii) Write program code to call the method `Learn()` for the character created in part **3(d)(i)**.

Output the name, age and intelligence of the character in an appropriate message.

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

[1]

BLANK PAGE

BLANK PAGE

BLANK PAGE