9618

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **May/June 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 [ ].

DC (LO) 214149

[Turn over

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 will 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.

A list is an alternative to an array.

A source file is used to answer question 3. The file is called **`TreasureChestData.txt`**

1 An unordered linked list uses a 1D array to store the data.

Each item in the linked list is of a record type, `node`, with a field `data` and a field `nextNode`.

The current contents of the linked list are:

startPointer	0
emptyList	5

Index	**data**	**nextNode**
0	1	1
1	5	4
2	6	7
3	7	-1
4	2	2
5	0	6
6	0	8
7	56	3
8	0	9
9	0	-1

(a) The following is pseudocode for the record type `node`.

```
TYPE node
    DECLARE data : INTEGER
    DECLARE nextNode : INTEGER
ENDTYPE
```

Write program code to declare the record type `node`.

Save your program as **question1**.

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

[2]

(b) Write program code for the main program.

Declare a 1D array of type `node` with the identifier `linkedList`, and initialise it with the data shown in the table on page 2. Declare the pointers.

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

[4]

(c) The procedure `outputNodes()` takes the array and `startPointer` as parameters. The procedure outputs the data from the linked list by following the `nextNode` values.

(i) Write program code for the procedure `outputNodes()`.

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

[6]

(ii) Edit the main program to call the procedure `outputNodes()`.

Take a screenshot to show the output of the procedure `outputNodes()`.

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

[1]

(d) The function, `addNode()`, takes the linked list and pointers as parameters, then takes as input the data to be added to the end of the `linkedList`.

The function adds the node in the next available space, updates the pointers and returns `True`. If there are no empty nodes, it returns `False`.

(i) Write program code for the function `addNode()`.

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

[7]

(ii) Edit the main program to:

- call `addNode()`
- output an appropriate message depending on the result returned from `addNode()`
- call `outputNodes()` twice; once before calling `addNode()` and once after calling `addNode()`.

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

[3]

(iii) Test your program by inputting the data value `5` and take a screenshot to show the output.

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

[1]

2 A program stores the following ten integers in a 1D array with the identifier `arrayData`.

10 5 6 7 1 12 13 15 21 8

(a) Write program code for a **new program** to:

- declare the global 1D array, `arrayData`, with ten elements
- initialise `arrayData` in the main program using the data values shown.

Save your program as **question2**.

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

[2]

(b) (i) A function, `linearSearch()`, takes an integer as a parameter and performs a linear search on `arrayData` to find the parameter value. It returns `True` if it was found and `False` if it was not found.

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

Save your program.

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

[6]

(ii) Edit the main program to:

- allow the user to input an integer value
- pass the value to `linearSearch()` as the parameter
- output an appropriate message to tell the user whether the search value was found or not.

Save your program.

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

[4]

(iii) Test your program with one value that is in the array and one value that is not in the array.

Take a screenshot to show the result of each test.

Save your program.

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

[2]

(c) The following bubble sort pseudocode algorithm sorts the data in `theArray` into descending numerical order. There are **five** incomplete statements.

```
PROCEDURE bubbleSort()

   DECLARE temp : INTEGER

   FOR x ← 0 to ……

        FOR y ← 0 to ……

            IF theArray[y] …… theArray[y + 1] THEN

               temp ← theArray[y]

               theArray[y] ← ……

               theArray[y + 1] ← ……

            ENDIF

        NEXT y

   NEXT x

ENDPROCEDURE
```

Write program code for the procedure `bubbleSort()` to sort the data in `arrayData` into descending order.

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

[6]

3 A computer game requires users to travel around a world to find and open treasure chests. Each treasure chest has a mathematics question inside. The user enters the answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

The program will be created using object-oriented programming (OOP).

The following class diagram describes the class `TreasureChest`.

TreasureChest	

`question : STRING`
`answer : INTEGER`
`points : INTEGER`
	`// stores the question`
	`// stores the answer`
	`// stores the maximum possible number of points available for this chest`

`constructor()`	`// takes question, answer and points as parameters and creates an instance of an object`
`getQuestion()`	`// returns the question`
`checkAnswer()`	`// takes the user's answer as a parameter and returns True if it is correct, otherwise returns False`
`getPoints()`	`// takes the number of attempts as a parameter and returns the number of points awarded`

(a) Create a new program.

Write program code to declare the class `TreasureChest`.

Do **not** write any other methods.

The attributes are private.

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

Save your program as **question3**.

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

[5]

(b) The text file `TreasureChestData.txt` stores data for five questions, in the order of question, answer, points.

For example, the first three lines of the file are for the first question:

```
2*2 question
4 answer
10 points
```

Write program code for the procedure, `readData()` to:

- read each question, answer and points from the text file
- create an object of type `TreasureChest` for each question
- declare an array named `arrayTreasure` of type `TreasureChest`
- append each object to the array
- use exception handling to output an appropriate message if the file is not found.

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

[8]

(c) The main program repeats each question until the user inputs the correct answer. The number of points awarded depends on the number of attempts before the user gives the correct answer.

(i) The class `TreasureChest` has a method `getQuestion()` that returns the question.

Write the method `getQuestion()`.

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

[1]

(ii) The class `TreasureChest` has a method `checkAnswer()` that takes the user's answer as a parameter. It returns `True` if the answer is correct and `False` otherwise.

Write the method `checkAnswer()`.

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

[3]

(iii) The class `TreasureChest` has a method `getPoints()` that takes the number of attempts as a parameter.

- If the number of attempts is 1, it returns the value of `points`.
- If the number of attempts is 2, it returns the integer value of `points` divided by 2 (`DIV 2`).
- If the number of attempts is 3 or 4, it returns the integer value of `points` divided by 4 (`DIV 4`).
- If the number of attempts is not 1 or 2 or 3 or 4, it returns `0` (zero).

For example, a question is worth 100 points and the user took 2 attempts to give the correct answer. The user is awarded 50 points (100 DIV 2).

Write the method `getPoints()`.

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

[5]

(iv) Write program code for the main program to:

- call the procedure `readData()`
- ask the user to enter a question number between 1 and 5
- output the question that matches the question number entered by the user
- check if the answer input by the user is correct using the method `checkAnswer()`
- repeat the question until the user inputs the correct answer
- count how many times the user attempted the question
- use the method `getPoints()` to return the number of points awarded
- output the number of points the user is awarded.

Save your program.

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

[7]

(v) Test the program.

Take a screenshot showing the input(s) and output(s) for each of the following two tests.

In the first test:

- select question 1 and answer it correctly the first time.

In the second test:

- select question 5 and answer it correctly the second time.

Save your program.

Copy and paste the screenshots into **part 3(c)(v)** in the evidence document.

[2]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **October/November 2021**

2 hours 30 minutes

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

5829148913

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]

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]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

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]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

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 [ ].

DC (KN) 303751/5

[Turn over

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 3**. The file is called **`CardValues.txt`**

1 A program needs to use a stack data structure. The stack can store up to 10 integer elements.

A 1D array `StackData` is used to store the stack globally. The global variable `StackPointer` points to the next available space in the stack and is initialised to 0.

(a) Write program code to declare the array and pointer as global data structures. Initialise the pointer to 0.

Save your program as **Question1_J22**. Copy and paste the program code into **part 1(a)** in the evidence document.

[3]

(b) Write a procedure to output all 10 elements in the stack **and** the value of `StackPointer`.

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

[3]

(c) The function `Push()` takes an integer parameter and returns `FALSE` if the stack is full. If the stack is not full, it puts the parameter value on the stack, updates the relevant pointer and returns `TRUE`.

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

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

[6]

(d) (i) Edit the main program to test the `Push()` function. The main program needs to:

- allow the user to enter 11 numbers and attempt to add these to the stack
- output an appropriate message when a number is added to the stack
- output an appropriate message when a number is not added to the stack if it is full
- output the contents of the stack after attempting to add all 11 numbers.

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

[5]

(ii) Test your program from **part 1(d)(i)** with the following 11 inputs:

11 12 13 14 15 16 17 18 19 20 21

Take a screenshot to show the output.

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

[1]

(e) The function `Pop()` returns `-1` if the stack is empty. If the stack is not empty, it returns the element at the top of the stack and updates the relevant pointer.

(i) Write program code for the function `Pop()`.

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

[5]

(ii) After the code you wrote in the main program for **part 1(d)(i)**, add program code to:

- remove two elements from the stack using `Pop()`
- output the updated contents of the stack.

Test your program and take a screenshot to show the output.

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

[2]

2 A 2D array stores data entered by a user.

(a) The main program declares a 2D array of 10 by 10 integer elements.

The array is initialised with a random number between 1 and 100 in each element.

Write program code for the main program.

Save your program as **Question2_J22**.
Copy and paste the program code into **part 2(a)** in the evidence document.

[4]

(b) The following bubble sort pseudocode algorithm sorts the data in the first dimension of the 2D array into ascending numerical order.

```
ArrayLength ← 10
FOR X ← 0 TO ArrayLength - 1
  FOR Y ← 0 TO ArrayLength - 2
    FOR Z ← 0 TO ArrayLength - Y - 2
      IF ArrayData[X, Z] > ArrayData[X, Z + 1] THEN
        TempValue ← ArrayData[X, Z]
        ArrayData[X, Z] ← ArrayData[X, Z+1]
        ArrayData[X, Z + 1] ← TempValue
      ENDIF
    NEXT Z
  NEXT Y
NEXT X
```

(i) Amend your main program by writing program code to implement the bubble sort algorithm after the initialisation of the array elements.

You must **not** use any built-in sorting functions for your programming language.

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

[5]

(ii) Write program code for a procedure to output all the values in the 2D array. The values should be output as a 2D grid, with values in rows and columns.

Call the procedure before and after your bubble sort code.

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

[3]

(iii) Test your program.

Take a screenshot to show the output.

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

[1]

(c) The following pseudocode function uses recursion to perform a binary search in the first row of the array, for the value `SearchValue` in the array `SearchArray`.

The function returns `-1` if the item was not found, or it returns the index where it is found.

There are **six** incomplete statements.

```
FUNCTION BinarySearch(SearchArray, Lower, Upper, SearchValue)RETURNS
                                                              INTEGER
  IF Upper >= Lower THEN
    Mid ← (Lower + (Upper - 1)) DIV ……
    IF SearchArray[0, Mid] = …… THEN
      RETURN ……
    ELSE
      IF SearchArray[0, Mid] > SearchValue THEN
        RETURN BinarySearch(SearchArray, ……, Mid - 1,
                                                    SearchValue)
      ELSE
        RETURN BinarySearch(SearchArray, Mid + 1, ……,
                                                    SearchValue)
      ENDIF
    ENDIF
  ENDIF
  RETURN ……
ENDFUNCTION
```

Note: the arithmetic operator `DIV` performs integer division, e.g. the result of `10 DIV 3` will be `3`.

(i) Write program code for the recursive function `BinarySearch()`.

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

[8]

(ii) In the main program, test the function `BinarySearch()` twice, outputting the returned value each time.

One test should be for a number that is in the first line of the array.
One test should be for a number that is not in the first line of the array.

Take a screenshot to show the output.

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

[2]

3 A programmer is designing a computer game that uses a set of cards.

Each card has a number and a colour. The cards are saved in the text file `CardValues.txt`

The program has a class named `Card`. The class has the following attributes and methods.

Card	
`Number : INTEGER`	The number of the card
`Colour : STRING`	The colour of the card
`Constructor()`	Takes two values as parameters and sets them to the private attributes
`GetNumber()`	Returns the number of the card
`GetColour()`	Returns the colour of the card

(a) The constructor takes the number and colour of the card as parameters and sets them to the private attributes.

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

Use your programming language appropriate constructor.

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

Save your program as **Question3_J22**.

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

[5]

(b) The two get methods return the associated attribute.

Write program code for the get methods `GetNumber()` and `GetColour()`.

Save your program.

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

[3]

(c) The text file `CardValues.txt` stores the data for 30 cards, in the order: number, colour.

For example, the first card in the text file:

`1` is the number
`red` is the colour.

A 1D array of type `Card` is declared to store all the cards read in from `CardValues.txt`

Write the main program to:

- declare an array of type `Card` with 30 elements
- read in the data for the 30 cards from `CardValues.txt` and assign each to the array.

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

[7]

(d) The program needs to allow all players (maximum of 5) to select 4 cards from the 30 available. A card can only be selected once, so the program needs to record which cards have already been selected.

The function, `ChooseCard()`:

- takes as input an integer to represent an array index from 1 to 30
- validates that the value is between 1 and 30 inclusive
- checks if the card is available (it has not already been selected)
- loops until an available card is selected
- returns the index of the card if it is available.

Amend the program to store which cards have already been selected **and** write program code for the function `ChooseCard()`.

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

[6]

(e) The main program needs to allow one player to select all their 4 cards.

(i) Amend the main program to:

- create an array, `Player1`, for player 1 of type `Card`
- ask player 1 to input 4 integers using the function from **part 3(d)**
- store the cards in `Player1`
- output the number and colour of the 4 cards in `Player1`.

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

[5]

(ii) Test your program with the following test data:

Test 1: **1** **5** **9** **10**

Test 2: **2** **2** **3** **4** **4** **5**

Take a screenshot to show the output.

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

[1]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **October/November 2022**

2 hours 30 minutes

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

0549214835

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_999**

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 1**. The file is called **`IntegerData.txt`**

1 The text file `IntegerData.txt` stores 100 integer numbers between 1 and 100 inclusive. A program is required to read in this data and perform searching and sorting on the data.

(a) Write program code to declare a global 1D array, `DataArray`, with space for 100 integer values.

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

[2]

(b) The procedure `ReadFile()` must read in the numbers from the text file and store each one in the array. Use appropriate exception handling.

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

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

[6]

(c) The function `FindValues()` asks the user to enter a number to search for in the array. The number input must be a whole number between 1 and 100 inclusive. The function then returns the number of times the number input appears in the array.

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

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

[7]

(d) (i) Write program code to call `ReadFile()` and `FindValues()` from the main program. The return value from `FindValues()` must be output with an appropriate message.

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

[3]

(ii) Test your program using the number 61 as input.

Take a screenshot to show the output.

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

[1]

(e) The procedure `BubbleSort()` needs to perform a bubble sort on the array and print the contents of the sorted array.

Write program code for the procedure `BubbleSort()` **and** call it from the main program.

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

[4]

2 A computer program is being developed that uses a set of cards. The program is written using object-oriented programming.

The program has two classes: `Card` and `Hand`.

The methods and attributes of these classes are shown:

Card	
`Number : INTEGER`	stores the card number from 1 to 5 inclusive
`Colour : STRING`	stores the card colour: red, blue or yellow
`Constructor()`	takes a number and colour as parameters and sets the private values to these parameters
`GetNumber()`	returns the card number
`GetColour()`	returns the card colour

Hand	
`Cards : ARRAY[0:9] OF Card`	1D array of type `Card`
`FirstCard : INTEGER`	stores the position of the first card in the hand
`NumberCards : INTEGER`	stores the number of cards in the hand
`Constructor()`	takes five card objects as parameters, assigns each card to the array `Cards[]`, initialises `FirstCard` to 0 and `NumberCards` to 5
`GetCard()`	takes an index as a parameter and returns the card at that index in the array

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

Do **not** write program code for the get 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)(i)** in the evidence document.

[5]

(ii) Write program code for the class methods `GetNumber()` and `GetColour()`.

Save your program.

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

[3]

(iii) The program is tested with the following cards:

Number	Colour
1	red
2	red
3	red
4	red
5	red
1	blue
2	blue
3	blue
4	blue
5	blue
1	yellow
2	yellow
3	yellow
4	yellow
5	yellow

Write program code to declare each of these cards as a variable of type `Card` in the main program.

Save your program.

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

[2]

(b) (i) Write program code to declare the class `Hand`, its attributes and constructor.

Do **not** write the get 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.

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

[6]

(ii) The get method `GetCard()` takes an index as a parameter and returns the card stored at that index in the array.

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

Save your program.

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

[2]

(iii) Two players are declared with 5 cards each.

Player 1 has the cards: 1 red, 2 red, 3 red, 4 red, 1 yellow.
Player 2 has the cards: 2 yellow, 3 yellow, 4 yellow, 5 yellow, 1 blue.

Write program code to declare player 1 and player 2 as objects of type `Hand`, with the cards indicated.

Save your program.

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

[2]

(c) The function `CalculateValue()` takes a player's hand as a parameter and returns a score calculated using the following rules:

- If a card is red, 5 points are added to the player's score.
- If a card is blue, 10 points are added to the player's score.
- If a card is yellow, 15 points are added to the player's score.
- The number of each card in the hand is added to the player's score.

(i) Write program code for the function `CalculateValue()`.
Assume that there are only 5 cards in the player's hand in this function.

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

[6]

(ii) Amend the main program by writing program code to use the function `CalculateValue()` for each of the two players. The player with the highest score wins.

Output an appropriate message to identify the winning player, or if the game was a draw (both players have the same number of points).

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

[4]

(iii) Test your program.

Take a screenshot to show the output.

Copy and paste the screenshot into **part 2(c)(iii)** in the evidence document.

[1]

3 A binary tree consists of nodes. Each node has 3 integer values: a left pointer, data and a right pointer.

The binary tree is stored using a global 2D array.

The pseudocode declaration for the array is:

```
DECLARE ArrayNodes : ARRAY[0:19, 0:2] OF INTEGER
```

For example:

- `ArrayNodes[0, 0]` stores the left pointer for the first node.
- `ArrayNodes[0, 1]` stores the data for the first node.
- `ArrayNodes[0, 2]` stores the right pointer for the first node.

`-1` indicates a null pointer, or null data.

(a) Write program code to:

- declare the global 2D array `ArrayNodes`
- initialise all 3 integer values to `-1` for each node.

Save your program as **Question3_N22**.

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

[3]

(b) The binary tree stores the following values:

Index	Left pointer	Data	Right pointer
0	1	20	5
1	2	15	–1
2	–1	3	3
3	–1	9	4
4	–1	10	–1
5	–1	58	–1
6	–1	–1	–1

`FreeNode` stores the index of the first free element in the array, initialised to 6.

`RootPointer` stores the index of the first node in the tree, initialised to 0.

Amend your program by writing program code to store the given data in `ArrayNodes` **and** initialise the free node and root node pointers.

Save your program.

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

[2]

(c) The following recursive pseudocode function searches the binary tree for a given value. If the value is found, the function must return the index of the value. If the value is not found, the function must return –1.

The function is incomplete. There are **four** incomplete statements.

```
FUNCTION SearchValue(Root : INTEGER,
                     ValueToFind : INTEGER) RETURNS INTEGER

   IF Root = -1 THEN

      RETURN -1

   ELSE

     IF ArrayNodes[Root, 1] = ValueToFind THEN

        RETURN .....

     ELSE

        IF ArrayNodes[Root, 1] = -1 THEN

           RETURN -1

        ENDIF

     ENDIF

   ENDIF

   IF ArrayNodes[Root, 1] ..... ValueToFind THEN

      RETURN SearchValue(ArrayNodes[....., 0], ValueToFind)

   ENDIF

   IF ArrayNodes[Root, .....] < ValueToFind THEN

      RETURN SearchValue(ArrayNodes[Root, 2], ValueToFind)

   ENDIF

ENDFUNCTION
```

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

Save your program.

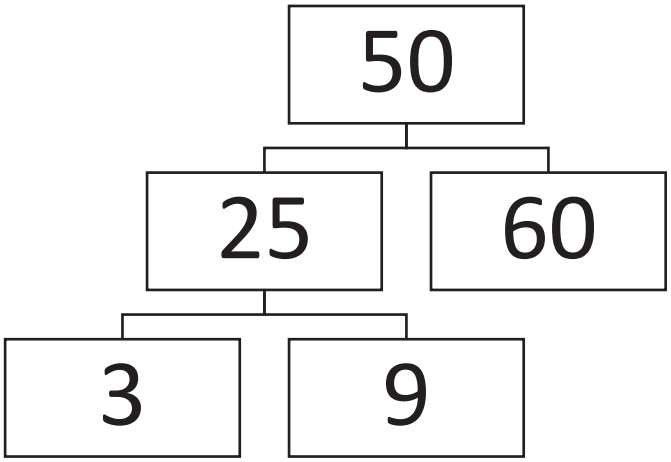
Copy and paste the program code into **part 3(c)** in the evidence document.

[5]

(d) A post order traversal performs the following operation:

- visit the left node
- visit the right node
- output the root.

For example, in the following tree, the output would be: **3 9 25 60 50**



An outline of the `PostOrder()` procedure is:

- If left node is not empty, make a recursive call with the left node as the root.
- If right node is not empty, make a recursive call with the right node as the root.
- Output the current root node.

The procedure `PostOrder()` takes the root node as a parameter.

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

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

[7]

(e) (i) Amend the main program by writing program code to:

- call the function `SearchValue()` to find the position of the number 15 in the tree
- use the result from `SearchValue()` to output either the index of the value if found, or an appropriate message to state that the value was not found
- call the procedure `PostOrder()`.

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

[3]

(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]

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]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **May/June 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 [ ].

DC (LK) 327187

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]

[Turn over

(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]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **May/June 2023**

2 hours 30 minutes

4774742619

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.

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

Two source files are used to answer **Question 3**. The files are called **`Employees.txt`** and **`HoursWeek1.txt`**

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.

1 A 1D array needs to store the names of 10 animals.

(a) Write program code to declare the global string array `Animals` to store 10 items.

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

[2]

(b) The main program needs to store the following animals in the array:

horse
lion
rabbit
mouse
bird
deer
whale
elephant
kangaroo
tiger

Write program code to store these animal names in the array.
The names must be in lower case and stored in the order given in the list.

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

[2]

(c) The following pseudocode procedure sorts the array into a descending alphabetical order using only the first character in each animal name.

The function `LENGTH(DataArray)` returns the number of elements in the array `DataArray`.

The function `MID(String, Start, Quantity)` returns `Quantity` number of characters from `String` starting at index `Start`. The first character in the string is index 0, for example:

`MID("tiger", 0, 2)` will return `"ti"`

There are **four** incomplete statements in the procedure.

```
PROCEDURE SortDescending()

  DECLARE ArrayLength : INTEGER

  DECLARE Temp : STRING

  ArrayLength ← LENGTH(Animals)

  FOR X ← 0 TO ArrayLength - 1

    FOR Y ← …… TO ArrayLength - X - 1

      IF MID(Animals[Y], 0, 1) < MID(Animals[……], 0, 1) THEN

        Temp ← Animals[……]

        Animals[Y] ← Animals[Y + 1]

        Animals[Y + 1] ← ……

      ENDIF

    NEXT Y

  NEXT X

ENDPROCEDURE
```

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

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

[6]

(d) (i) Write program code to amend the main program to:

- call the procedure `SortDescending()`
- output the sorted contents of the array with each animal name on a new line.

Save your program.

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

[3]

(ii) Test your program.

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 business sells a single product. Customers can purchase one or more of this product.

Each sale has an ID and a quantity, for example "ABC" and 2

The business needs a program to store the data about the sales in a circular queue data structure.

(a) Write program code to declare a record structure, `SaleData`, to store the data about each sale.

Save your program as **Question2_J2023**.

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

[2]

(b) Write program code to:

- declare a global array, `CircularQueue`, of 5 items to store the sale records
- declare the global pointers `Head` and `Tail`
- declare the global variable `NumberOfItems`
- initialise all elements of the array `CircularQueue` to an empty record, where the ID is null (`""`) and quantity is `-1`
- initialise `Head`, `Tail` and `NumberOfItems` to `0`

Save your program.

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

[4]

(c) The function `Enqueue()`:

- takes a new record as a parameter
- inserts the record in the circular queue at the element pointed to by `Tail`
- updates pointers and other variables as required
- returns `-1` if the circular queue is full
- returns `1` if the record is stored successfully.

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

Save your program.

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

[6]

(d) The function `Dequeue()`:

- returns a null or empty record if the circular queue is empty
- returns the first record in the queue if the circular queue is not empty
- updates pointers and other variables as required.

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

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

[6]

(e) The procedure `EnterRecord()`:

- takes an ID and quantity as input and creates a sale record
- uses `Enqueue()` to insert the record in the circular queue
- outputs `"Full"` if the record was not inserted in the circular queue
- outputs `"Stored"` if the record was inserted in the circular queue.

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

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

[5]

(f) The following sale records need to be entered into the program:

ID	Quantity
ADF	10
OOP	1
BXW	5
XXZ	22
HQR	6
LLP	3

(i) Amend the main program to:

- use `EnterRecord()` to input the six records in the table
- use `Dequeue()` to remove one record
- output either the ID and quantity of the removed record, or an error message if the circular queue is empty
- use `EnterRecord()` to input the record with the ID "LLP" for a second time
- output the ID and quantity for all the records currently stored in the array `CircularQueue`.

Write program code to perform these tasks.

Save your program.

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

[4]

(ii) Test your program.

Take a screenshot of the output.

Save your program.

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

[1]

3 A company needs a computer program to store data about its employees.

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

The class `Employee` stores data about the employees. Each employee has an employee number, a job title and hourly pay rate. The class will also store the amount they are paid each week over a 52-week year in a 1D array.

Employee	
`HourlyPay : REAL`	stores the amount each employee gets paid each hour
`EmployeeNumber : STRING`	stores the employee's unique number
`JobTitle : STRING`	stores the employee's job title
`PayYear2022 : ARRAY[0:51] OF REAL`	stores the amount the employee has been paid each week
`Constructor()`	initialises `HourlyPay`, `EmployeeNumber` and `JobTitle` from the values passed as parameters initialises all 52 elements in `PayYear2022` to `0.0`
`GetEmployeeNumber()`	returns the employee number
`SetPay()`	takes the week number and number of hours worked that week as parameters calculates and stores the pay for that week in `PayYear2022`
`GetTotalPay()`	returns the total of all the values in `PayYear2022`

(a) (i) Write program code to declare the class `Employee`.

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

Use your programming language appropriate constructor.

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

Save your program as **Question3_J2023**.

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

[5]

(ii) The method `GetEmployeeNumber()` returns the employee number.

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

Save your program.

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

[2]

(iii) The method `SetPay()`:

- takes a week number and the number of hours worked that week as parameters
- calculates the pay for that week by multiplying the hourly pay by the number of hours worked that week
- stores the calculated pay in the appropriate index for that week in `PayYear2022`.

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

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

[3]

(iv) The method `GetTotalPay()` returns the total of all the values in `PayYear2022`.

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

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

[2]

(b) The child class `Manager` inherits from the parent class `Employee`.

A manager gets a bonus. This bonus value is a percentage, for example 10.0%.
When calculating the pay, the number of hours the manager worked that week is increased by the bonus value.

Manager	
`BonusValue : REAL`	stores the bonus value, for example 10.0 represents a 10.0% increase
`Constructor()`	takes bonus value, hourly pay, employee number and job title as parameters initialises `BonusValue` to its parameter value
`SetPay()`	takes the week number and number of hours worked as parameters increases the number of hours worked by the bonus value calls the `SetPay()` method from the parent class

(i) Write program code to declare the class `Manager`.

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

Use your programming language 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(b)(i)** in the evidence document.

[4]

(ii) The `Manager` method `SetPay()` overrides the method from the parent class and:

- takes the week number and number of hours worked as parameters
- increases the number of hours worked by the bonus value
- calls `SetPay()` from the parent class.

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

Save your program.

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

[3]

(c) The main program has a global 1D array, `EmployeeArray`, to store data about eight employees. Each employee is stored as an object of type `Employee`.

The file `Employees.txt` stores data about the employees, in the order:

- hourly pay
- employee number
- bonus value (where included)
- job title.

Only employees who are managers have a bonus value saved. For example:

- The first employee is a Junior Developer, with employee number 12452 and an hourly pay of $15.22. This employee does not have a bonus value.
- The third employee is an Interface Manager, with employee number 02586 and an hourly pay of $22.50. This employee has a bonus value of 5.25%.

Write the main program to:

- declare the array to store data about 8 employees
- read in the data from the file for each employee
- instantiate each employee as either `Employee` (if the employee does not have a bonus value) or `Manager` (if the employee has a bonus value).

Save your program.

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

[7]

(d) The file `HoursWeek1.txt` stores the number of hours each employee has worked in week 1, in the order:

- employee number
- number of hours worked.

For example, the first set of data is for employee 21548 who has worked 50.0 hours.

The procedure `EnterHours()`:

- reads in the values from the file
- finds the location of each employee in `EmployeeArray`
- calls the method `SetPay()` for each employee.

Write program code for `EnterHours()`.

Save your program.

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

[4]

(e) (i) The main program needs to call `EnterHours()` and use the method `GetTotalPay()` to output the employee number and total pay for each of the eight employees.

Amend the main program to perform these tasks.

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

[2]

(ii) Test your program.

Take a screenshot of the output.

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

[1]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/41**

Paper 4 Practical **October/November 2023**

2 hours 30 minutes

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

8815737793

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 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 2**. The file is called **`QueueData.txt`**

1 This iterative pseudocode algorithm for the function `IterativeVowels()` takes a string as a parameter and counts the number of lower-case vowels in this string.
The vowels are the letters a, e, i, o and u.

```
FUNCTION IterativeVowels(Value : STRING) RETURNS INTEGER
   DECLARE Total : INTEGER
   DECLARE LengthString : INTEGER
   DECLARE FirstCharacter : CHAR
   Total ← 0
   LengthString ← LENGTH(Value)
   FOR X ← 0 TO LengthString - 1
      FirstCharacter ← MID(Value, 0, 1)
      IF FirstCharacter = 'a' OR FirstCharacter = 'e' OR
         FirstCharacter = 'i' OR FirstCharacter = 'o' OR
         FirstCharacter = 'u' THEN
            Total ← Total + 1
      ENDIF
      Value ← MID(Value, 1, LENGTH(Value)-1)
   NEXT X
   RETURN Total
ENDFUNCTION
```

The pseudocode function `MID(X, Y, Z)` returns `Z` number of characters from string `X`, starting at the character in position `Y`. The first character in a string is in position 0, for example:

`MID("computer", 0, 3)` returns `"com"`

The pseudocode function `LENGTH(X)` returns the number of characters in the string `X`, for example:

`LENGTH("computer")` returns 8

(a) (i) Write program code for the function `IterativeVowels()`.

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

[5]

(ii) Write program code to call the function `IterativeVowels()` with the parameter `"house"` from the main program.

Output the return value.

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

[1]

(b) (i) Rewrite the function `IterativeVowels()` as a recursive function with the identifier `RecursiveVowels()`.

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

[6]

(ii) Write program code to call the function `RecursiveVowels()` with the parameter `"imagine"` from the main program.

Output the return value.

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]

2 A linear queue is implemented using the 1D array, `Queue`. The index of the first element in the array is 0.

(a) (i) Write program code to declare:

- `Queue` — a global array with space to store 50 IDs of type string
- `HeadPointer` — a global variable to point to the first element in the queue, initialised to −1
- `TailPointer` — a global variable to point to the next available space in the queue, initialised to 0.

Save your program as **Question2_N23**.

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

[2]

(ii) The procedure `Enqueue()` takes a string parameter.

If the queue is full, the procedure outputs a suitable message. If the queue is not full, the procedure inserts the parameter into the queue and updates the relevant pointer(s).

Write program code for `Enqueue()`.

Save your program.

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

[4]

(iii) The function `Dequeue()` checks if the queue is empty.

If the queue is empty, the function outputs a suitable message and returns the string `"Empty"`.
If the queue is not empty, the function returns the first element in the queue and updates the relevant pointer(s).

Write program code for `Dequeue()`.

Save your program.

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

[4]

(b) A shop sells computer games. Each game has a unique identifier (ID) of string data type.

The text file `QueueData.txt` contains a list of game IDs.

The procedure `ReadData()` reads the data from the text file and inserts each item of data into the array `Queue`.

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

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

[6]

(c) Some game IDs appear in the text file more than once.

The program needs to total the number of times each game ID appears in the text file.

The record structure `RecordData` has the following fields:

- `ID` — a string to store the game ID
- `Total` — an integer to store the total number of times that game ID appears in the text file.

(i) Write program code to declare the record structure `RecordData`.

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

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

[2]

(ii) The global 1D array `Records` stores up to 50 items of type `RecordData`.

The global variable `NumberRecords` stores the number of records currently in the array `Records` and is initialised to 0.

Write program code to declare `Records` and `NumberRecords`.

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

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

[2]

(iii) The pseudocode algorithm for the procedure `TotalData()`:

- uses `Dequeue()` to remove an ID from the queue
- checks whether a `RecordData` with the returned ID exists in `Records`
- increments the total for that ID in the record if the ID already exists
- creates a new record and stores it in `Records` if the ID does not exist.

```
PROCEDURE TotalData()
   DECLARE DataAccessed : STRING
   DECLARE Flag : BOOLEAN
   DataAccessed ← Dequeue()
   Flag ← FALSE
   IF NumberRecords = 0 THEN
      Records[NumberRecords].ID ← DataAccessed
      Records[NumberRecords].Total ← 1
      Flag ← TRUE
      NumberRecords ← NumberRecords + 1
    ELSE
      FOR X ← 0 TO NumberRecords - 1
         IF Records[X].ID = DataAccessed THEN
            Records[X].Total ← Records[X].Total + 1
            Flag ← TRUE
         ENDIF
       NEXT X
   ENDIF
   IF Flag = FALSE THEN
      Records[NumberRecords].ID ← DataAccessed
      Records[NumberRecords].Total ← 1
      NumberRecords ← NumberRecords + 1
   ENDIF
ENDPROCEDURE
```

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

Save your program.

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

[5]

(d) The procedure `OutputRecords()` outputs the ID and total of each record in `Records` in the format:

```
ID  1234   Total   4
```

Write program code for `OutputRecords()`.

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

[1]

(e) The main program needs to:

- call `ReadData()`
- call `TotalData()` for each element in the queue
- call `OutputRecords()`.

(i) Write program code for the main program.

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

[2]

(ii) Test your program.

Take a screenshot of the output.

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

[1]

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

The game has multiple characters that can move around the screen.

The class `Character` stores data about the characters. Each character has a name, a current X (horizontal) position and a current Y (vertical) position.

Character	
`Name : STRING`	stores the name of the character as a string
`XPosition : INTEGER`	stores the X position as an integer
`YPosition : INTEGER`	stores the Y position as an integer
`Constructor()`	initialises `Name`, `XPosition` and `YPosition` to its parameter values
`GetXPosition()`	returns the X position
`GetYPosition()`	returns the Y position
`SetXPosition()`	adds the parameter to the X position validates that the new X position is between 0 and 10000 inclusive
`SetYPosition()`	adds the parameter to the Y position validates that the new Y position is between 0 and 10000 inclusive
`Move()`	takes a direction as a parameter and calls either `SetXPosition` or `SetYPosition` with an integer value

(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.

[4]

(ii) The get methods `GetXPosition()` and `GetYPosition()` each return the relevant attribute.

Write program code for the get methods.

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

[3]

(iii) The set methods `SetXPosition()` and `SetYPosition()` each take a value as a parameter and add this to the current X or Y position.

If the new value exceeds 10 000, it is limited to 10 000.

If the new value is below 0, it is limited to 0.

Write program code for the set methods.

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

[4]

(iv) The method `Move()` takes a string parameter: `"up"`, `"down"`, `"left"` or `"right"`.

The table shows the change each direction will make to the X or Y position.

Use the appropriate method to change the position value.

Direction	Value change
up	Y position + 10
down	Y position − 10
left	X position − 10
right	X position + 10

Write program code for `Move()`.

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

[4]

(b) Write program code to declare a new instance of `Character` with the identifier `Jack`.

The starting X position is 50 and the starting Y position is 50, the character's name is Jack.

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

[2]

(c) The class `BikeCharacter` inherits from the class `Character`.

BikeCharacter	
`Constructor()`	takes `Name`, `XPosition` and `YPosition` as parameters calls its parent class constructor with the appropriate values
`Move()`	overrides the method `Move()` from the parent class by changing either the X position or the Y position by 20 instead of 10

(i) Write program code to declare the class `BikeCharacter` 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.

[3]

(ii) The method `Move()` overrides the method from the parent class.

The table shows the change each direction will make to the X or Y position.

Direction	Value change
up	Y position + 20
down	Y position − 20
left	X position − 20
right	X position + 20

Write program code for `Move()`.

Save your program.

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

[2]

(d) Write program code to declare a new instance of `BikeCharacter` with the identifier `Karla`.

The starting X position is 100, the starting Y position is 50 and the character's name is Karla.

Save your program.

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

[1]

(e) (i) Write program code to:

- take as input which of the two characters the user would like to move
- take as input the direction the user would like the character to move
- call the appropriate method to move the character
- output the character's new X and Y position in an appropriate format, for example:

```
"Karla's new position is X = 100 Y = 200"
```

All inputs require appropriate prompts and must be validated.

Save your program.

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

[5]

(ii) Test your program twice with the following inputs.

Test 1: jack right

Test 2: karla down

Take a screenshot of the output.

Save your program.

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

[2]

Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **October/November 2023**

2 hours 30 minutes

9002499716

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]

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]

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]