Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/43**

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 [ ].

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 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]

BLANK PAGE

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]

BLANK PAGE