Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/42**

Paper 4 Practical **May/June 2025**

2 hours 30 minutes

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

0177211580

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.

You have been supplied with the following source files:

`StackData.txt`
`SecondStack.txt`
`HashData.txt`

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 used does not support arrays, a list can be used instead.

Three source files are used to answer **Questions 1** and **2**. The files are called **`StackData.txt`**, **`SecondStack.txt`** and **`HashData.txt`**

1 A program reads data from a text file and stores it in a stack. The stack `Stack` is stored as a 1D array of up to 20 elements. The pointer `TopOfStack` stores the index of the last element stored in the stack.

(a) `Stack` is a global array of strings with all elements initialised to `"-1"`

`TopOfStack` is a global variable initialised to `-1`

Write program code to declare and initialise `Stack` and `TopOfStack`

Save your program as **Question1_J25**.

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

[2]

(b) The function `Push()` takes a string parameter and attempts to store it on the stack.

The function returns `-1` if the stack is full.

The function returns `1` if the parameter is successfully pushed onto the stack.

Write program code for `Push()`

Save your program.

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

[4]

(c) The function `Pop()` returns the next item from the stack.

The function returns `"-1"` if the stack is empty.

Write program code for `Pop()`

Save your program.

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

[4]

(d) The text file `StackData.txt` stores numbers and mathematical operators. Each number and operator are on a new line in the text file.

The mathematical operators used in this program are:

mathematical operator	**meaning**
+	addition
–	subtraction
/	division
*	multiplication
^	power of

The procedure `ReadData()`:

- takes a string filename as a parameter
- opens the file and reads in each line of data
- uses the `Push()` function to insert each line of data onto the stack
- outputs `"Stack full"` if any data cannot be stored on the stack because the stack is full
- uses exception handling when opening and reading from the text file.

The procedure needs to work for a file that contains an unknown number of lines.

Write program code for `ReadData()`

Save your program.

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

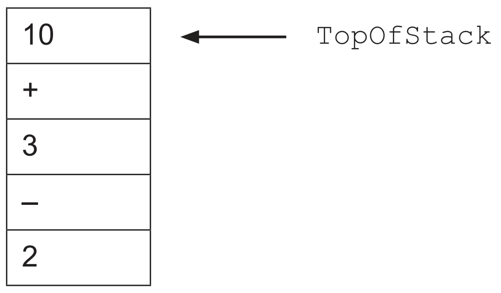
[6]

(e) The values stored in the stack are used to perform mathematical operations.

A total is initialised to the first value in the stack. The first value in the stack will always be a number.

The next value will be a mathematical operator and the next value will be the number to apply to the calculation. This is repeated until there are no values left in the stack. There will always be a number following a mathematical operator.

For example, the contents of a stack are:



The total is initialised to the first value in the stack: 10

total = 10

The next two values are: + 3

total = 10 + 3 = 13

The next two values are: – 2

total = 13 – 2 = 11

The final total is 11 and this is returned.

Write program code for the function `Calculate()` to:

- take each value from the stack
- calculate and return the final total.

Save your program.

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

[7]

(f) **(i)** Write program code to extend the main program to:

- take a filename as input from the user
- call `ReadData()` with the input filename
- call `Calculate()`
- output the final total.

Save your program.

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

[2]

(ii) Test your program twice with the file names:

```
StackData.txt
SecondStack.txt
```

Take a screenshot of the output(s).

Save your program.

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

[2]

2 A program stores data in a 1D array of records, `HashTable`. The array has space for 200 records. Each data item is stored in a specific index of the array that is calculated using an algorithm. The index is calculated from the key field using the formula: `Key MOD 200`

If two key fields generate the same index, there is a collision. Any records that have a collision are stored in a second array, `Spare`. This array has space for 100 records. When a collision is detected, the record is stored in the next free space in `Spare`.

(a) The pseudocode record format is:

```
TYPE NewRecord
  DECLARE Key : INTEGER
  DECLARE Item1 : INTEGER
  DECLARE Item2 : INTEGER
ENDTYPE
```

Write program code to declare the record type `NewRecord`

If your chosen programming language does **not** support record formats, a class can be used instead.

Save your program as **Question2_J25**.

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

[2]

(b) **(i)** Write program code to declare the global arrays `HashTable` and `Spare`

Save your program.

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

[1]

(ii) An empty record has the integer `−1` stored in each field.

The procedure `Initialise()` stores an empty record in each element in `HashTable` and `Spare`

Write program code for `Initialise()`

Save your program.

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

[2]

(c) The hash value is calculated from the key field using the formula: `Key MOD 200`

The function `CalculateHash()` takes a key field as a parameter and calculates and returns the hash value for the key field.

Write program code for `CalculateHash()`

Save your program.

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

[2]

(d) The procedure `InsertIntoHash()`:

- takes a record of type `NewRecord` as a parameter
- uses `CalculateHash()` to calculate the hash value for the key field in the record
- checks if the hash value index in `HashTable` currently stores an empty record
  - if the index stores an empty record, store the parameter in this index
  - if the index does **not** store an empty record, store the parameter in `Spare`

You can assume there will always be enough space in `Spare` to store any collisions.

Write program code for `InsertIntoHash()`

Save your program.

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

[6]

(e) The text file `HashData.txt` stores up to 200 rows of data to be stored into the hash table. Each row contains three integer numbers separated by commas.

The first number is the key field, the second number is item 1 and the third number is item 2.

For example:

The first row in the text file contains: `646`, `12`, `568`

The key field is `646`, item 1 is `12` and item 2 is `568`

The procedure `CreateHashTable()`:

- opens the file `HashData.txt`
- creates a record for each row of data in the file
- calls `InsertIntoHash()` with each record.

Write program code for `CreateHashTable()`

Save your program.

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

[5]

(f) **(i)** The procedure `PrintSpare()` outputs the key field of each element in the array `Spare` that does **not** contain an empty record.

Write program code for `PrintSpare()`

Save your program.

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

[2]

(ii) The main program should call the procedure to initialise the arrays, call the procedure to create the hash table and then call the procedure to output the contents of the array `Spare`

Write program code for the main program.

Save your program.

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

[1]

(iii) Test your program.

Take a screenshot of the output(s).

Save your program.

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

[1]

3 A program stores data about animals using Object-Oriented Programming (OOP).

The class `Animal` stores the data about animals.

`Animal`	
`Name : String`	stores the name of the animal
`Sound : String`	stores the sound the animal makes
`Size : Integer`	stores the size of the animal as an integer between 1 (smallest) and 10 (largest)
`Intelligence : Integer`	stores the intelligence of the animal as an integer between 1 (least) and 10 (most)
`Constructor()`	initialises all attributes to its parameter values
`Description()`	returns a string message that contains the data from the attributes

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

Do **not** declare the other methods.

Use your programming language appropriate constructor.

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

Save your program as **Question3_J25**.

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

[4]

(ii) The method `Description()` creates and returns a string of the animal's data in the format:

```
"The animal's name is " <Name> ", it makes a " <Sound> ", its size
is " <Size> " and its intelligence level is " <Intelligence>
```

For example:

```
The animal's name is Teddy, it makes a Bark, its size is 4 and its
intelligence level is 6
```

Write program code for `Description()`

Save your program.

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

[3]

(b) The class `Parrot` inherits from the class `Animal`

`Parrot` inherits the attributes from `Animal` and overrides the `Description()` method. The additional attributes and methods in the class `Parrot` are:

Parrot	
`WingSpan : Integer`	the width of the parrot's wings to the nearest cm
`NumberWords : Integer`	the number of words the parrot can speak
`Constructor()`	calls the parent constructor using its parameter values; initialises `WingSpan` and `NumberWords` to its parameter values
`ChangeNumberWords()`	takes an integer parameter and adds this to the number currently in `NumberWords`

(i) Write program code to declare the class `Parrot`, its constructor and the method `ChangeNumberWords()`

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 method `Description()` in the class `Parrot` creates and returns a string of the animal's data in the format:

```
"The animal's name is " <Name> ", it makes a " <Sound> ", its size
is " <Size> " and its intelligence level is " <Intelligence> ". It
has a wingspan of " <WingSpan> "cm and can say " <NumberWords> "
words."
```

For example:

```
The animal's name is Chewie, it makes a Squawk, its size is 1 and
its intelligence level is 10. It has a wingspan of 30cm and can
say 29 words.
```

Write program code for `Description()`

Save your program.

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

[2]

(c) The class `Wolf` inherits from the class `Animal`

`Wolf` inherits the attributes from `Animal` and overrides the `Description()` method. The additional attributes and methods in the class `Wolf` are:

`Wolf`	
`TerritorySize : Integer`	stores the size of the area where the wolf lives, to the nearest square mile
`Constructor()`	calls the parent constructor using its parameter values; initialises `TerritorySize` to its parameter value
`SetTerritorySize()`	takes an integer parameter and adds this to the number currently in `TerritorySize`

(i) Write program code to declare the class `Wolf`, its constructor and the method `SetTerritorySize()`

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(c)(i)** in the evidence document.

[4]

(ii) The method `Description()` in the class `Wolf` creates and returns a string of the animal's data in the format:

```
"The animal's name is " <Name> ", it makes a " <Sound> ", its size
is " <Size> " and its intelligence level is " <Intelligence> ".
Its territory is " <TerritorySize> " square miles."
```

For example:

```
The animal's name is Nighteyes, it makes a Howl, its size is 8 and
its intelligence level is 7. Its territory is 100 square miles.
```

Write program code for `Description()`

Save your program.

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

[2]

(d) **(i)** The main program declares instances of the classes for **three** animals:

- A parrot with the name ‘Chewie’; it makes a ‘Squawk’ sound. Its size is 1, intelligence is 10, wingspan is 30 cm and it can say 29 words.
- A wolf with the name ‘Nighteyes’; it makes a ‘Howl’ sound. Its size is 8, intelligence is 7 and its territory is 100 square miles.
- An animal that is a horse with the name ‘Copper’; it makes a ‘Neigh’ sound. Its size is 10 and its intelligence is 6.

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.

[2]

(ii) The main program also needs to:

- decrease the territory for the wolf Nighteyes by 20 square miles
- increase the number of words the parrot Chewie can say by 2 words
- output the description for all **three** animals.

Write program code to extend the main program.

Save your program.

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

[3]

(iii) Test your program.

Take a screenshot of the output(s).

Save your program.

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

[2]