Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	

CENTRE NUMBER						CANDIDATE NUMBER					

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2019**

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

Question 1 begins on the next page.

1 **(a)** **(i)** Programming languages can support different data types.

Complete the table by naming **three** different data types together with an example data value for each.

Data type	**Example data value**

[6]

(ii) Identify the type of programming statement that assigns a data type to a variable.

.... [1]

(b) As part of the development of an algorithm, a programmer may construct an identifier table.

Describe what an identifier table contains.

....

....

....

.... [2]

(c) **(i)** Simple algorithms usually consist of three different stages.

Complete the table below. Write each example statement in **program code**.

The second stage has already been given.

Stage	**Example statement**
Process	

[5]

(ii) Write a **single** statement in **program code** that contains **two** of the stages. Do **not** repeat any of the statements from **part (c)(i)**.

.... [1]

(d) A software developer is writing a program and includes several features to make it easier to read and understand. One of these features is the use of indentation.

State **three other** features.

Feature 1 ....

Feature 2 ....

Feature 3 ....
[3]

(e) A trace table is often used during program testing.

Identify the type of testing that includes the use of a trace table.

.... [1]

2 **(a)** **(i)** Two types of loop that may be found in an algorithm are the 'pre-condition' and 'post-condition' loop.

Identify **one other** type of loop. Explain when it should be used.

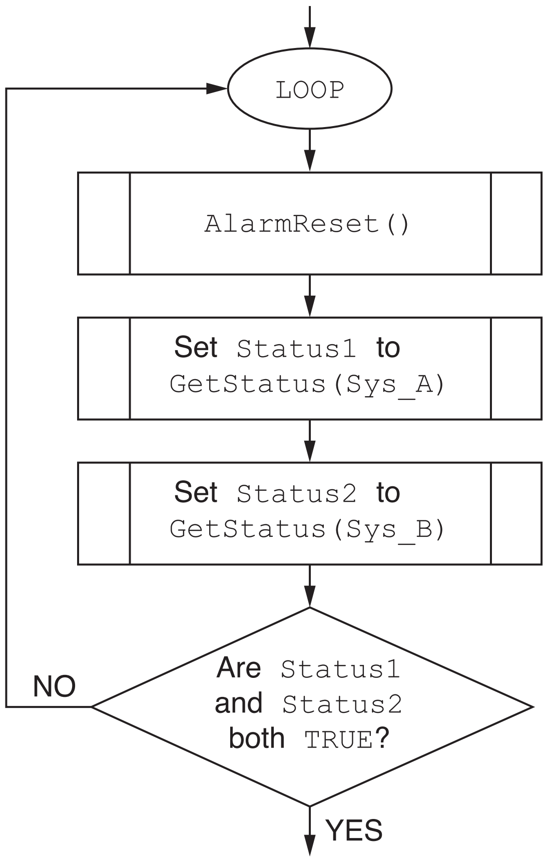
Type ....

Explanation ....

....

.... [2]

(ii) Part of a program flowchart is shown.



LOOP
AlarmReset()
Set Status1 to GetStatus(Sys_A)
Set Status2 to GetStatus(Sys_B)
Are Status1 and Status2 both TRUE?
NO
YES

Implement the flowchart in **pseudocode** using a post-condition loop.

....

....

....

....

....

.... [4]

(b) The following lines of code are taken from a high-level language program.

```
100 setvar(Count, Integer)
110 setvar(Gross[0-20], Real)
120 setvar(Posn, Real)
130 setvar(Length, Integer)
140 setvar(Rate, Real)
150 Length := 7
160 Rate := 1.175
170
180 For (Count, 0, 20, 2)
190  {
200    Echo "Input next cost"
210    Posn := Read()
220    Gross[Count] := Mult(Posn, Rate) %Apply current tax rate
230  }
```

Study the code. Identify the relevant features in the following table.

Feature	Answer
The symbol used to indicate an assignment	
The line numbers for the start and end of a count-controlled loop	
The step value of the count-controlled loop	
The character that indicates a comment	
The name of a function	

[5]

(c) A program written in a high-level language cannot be run directly.

Identify **one** type of translator that can be used to translate the program.

.... [1]

3 Three program modules process updating of passwords in a file. A description of the relationship between the modules is summarised as follows:

Module name	**Description**
`GetPassword()`	• Takes two parameters: `AccountID` and `OldPassword` • Returns a string containing the new password
`UpdateFile()`	• Takes two parameters: `AccountID` and `NewPassword` • Returns a Boolean value to indicate whether or not the update was successful
`ChangePassword()`	• Calls `GetPassword()` to obtain the new password then calls `UpdateFile()` to write the new password to the file

Draw a structure chart to show the relationship between the three modules and the parameters passed between them.

[5]

4 The following pseudocode algorithm checks whether a string is a valid email address.

```
FUNCTION Check(InString : STRING) RETURNS BOOLEAN

   DECLARE Index : INTEGER
   DECLARE NumDots : INTEGER
   DECLARE NumAts : INTEGER
   DECLARE NextChar : CHAR
   DECLARE NumOthers : INTEGER

   NumDots ← 0
   NumAts ← 0
   NumOthers ← 0

   FOR Index ← 1 TO LENGTH(InString)

      NextChar ← MID(InString, Index, 1)
      CASE OF NextChar
         '.': NumDots ← NumDots + 1
         '@': NumAts ← NumAts + 1
         OTHERWISE NumOthers ← NumOthers + 1
      ENDCASE

   ENDFOR

   IF (NumDots >= 1 AND NumAts = 1 AND NumOthers > 5)
      THEN
         RETURN TRUE
      ELSE
         RETURN FALSE
   ENDIF

ENDFUNCTION
```

(a) Describe the validation rules that are implemented by this pseudocode. Refer **only** to the contents of the string and **not** to features of the pseudocode.

....

....

....

.... [3]

(b) (i) Complete the trace table by dry running the function when it is called as follows:

```
Result ← Check("Jim.99@skail.com")
```

Index	NextChar	NumDots	NumAts	NumOthers

[5]

(ii) State the value returned when function `Check` is called as shown in **part (b)(i)**.

.... [1]

(c) The function `Check()` is to be tested.

State **two** different invalid string values that could be used to test the algorithm. Each string should test a different rule.

Justify your choices.

Value ....

Justification ....

....

....

Value ....

Justification ....

....

....
[4]

5 Abbreviations are often used in place of a full name. Concatenating the first letter of each word in the name makes an abbreviation.

For example:

Name	**Abbreviation**
United Nations	UN
World Wide Web	WWW
British Computer Society	BCS

A function, `Abbreviate()`, will take a string representing the full name and return a string containing the abbreviated form.

You should assume that:

- names only contain alphabetic characters and space characters
- names always start with an alphabetic character
- each word in the name always starts with an uppercase character
- only a single space separates words in the name.

Write **pseudocode** to implement the function `Abbreviate()`.

Refer to the **Appendix** on page 16 for the list of built-in functions and operators.

[8]

6 A text file, `Library.txt`, stores information relating to a book collection. The file stores four pieces of information about each book on separate lines of the file, as follows:

```
Line n:       <Book Title>
Line n + 1:   <Author Name>
Line n + 2:   <ISBN>
Line n + 3:   <Location>
```

Information is stored as data strings.

Information relating to two books is shown:

File line	Data
100	"Learning Python"
101	"Brian Smith"
102	"978-14-56543-21-8"
103	"BD345"
104	"Surviving in the mountains"
105	"C T Snow"
106	"978-35-17635-43-9"
107	"ZX001"

(a) **(i)** A function, `FindBooksBy()`, will search `Library.txt` for all books by a given author.

The function will store the `Book Title` and `Location` in the array `Result`, and will return a count of the number of books found.

Array `Result` is a global 2D array of type `STRING`. It has 100 rows and 2 columns.

Write **pseudocode** to declare the array `Result`.

....

....

.... [3]

(ii) Function `FindBooksBy()` will:

- receive the `Author Name` as a parameter
- search `Library.txt` for matching entries
- store the `Book Title` and `Location` of matching entries in the `Result` array
- return an integer value giving the number of books by the author that were found.

Write **program code** for the function `FindBooksBy()`.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(b) The function `FindBooksBy()` has already been called and has stored values in the array `Result`.

The procedure, `DisplayResults()`, will output the information from the array.

The procedure receives the following two parameters:

- a string containing the author name
- an integer value representing the number of books found

The output should be formatted as in the following example:

```
Books written by: Brian Smith

Title                     Location
Learning Python           BD345
Arrays are not lists      CZ562
Learning Java             CZ589

Number of titles found: 3
```

If no books by the author are found, the following should be output:

```
Search found no books by: Brian Smith
```

Write **pseudocode** for the procedure `DisplayResults()`.

Refer to the **Appendix** on page 16 for the list of built-in functions and operators.

[7]

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

```
MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING
```
returns a string of length `y` starting at position `x` from `ThisString`

Example: `MID("ABCDEFGH", 2, 3)` returns `"BCD"`

```
LENGTH(ThisString : STRING) RETURNS INTEGER
```
returns the integer value representing the length of `ThisString`

Example: `LENGTH("Happy Days")` returns `10`

```
LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING
```
returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns `"ABC"`

```
RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING
```
returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 3)` returns `"FGH"`

```
INT(x : REAL) RETURNS INTEGER
```
returns the integer part of `x`

Example: `INT(27.5415)` returns `27`

```
ASC(ThisChar : CHAR) RETURNS INTEGER
```
returns the ASCII value of `ThisChar`

Example: `ASC('A')` returns `65`

```
MOD(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER
```
returns the integer value representing the remainder when `ThisNum` is divided by `ThisDiv`

Example: `MOD(10,3)` returns `1`

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` produces `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` produces `TRUE`