Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME									
CENTRE NUMBER						CANDIDATE NUMBER			

4420502994

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2018**

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.

CAMBRIDGE International Examinations

[Turn over

Question 1 begins on the next page.

1 **(a)** The following table contains statements written in pseudocode.

Show the type of programming construct each statement represents.

Put a tick (✓) in the appropriate column for each statement.

Statement	Assignment	Selection	Repetition (Iteration)
`CASE OF TempSensor1`			
`ELSE`			
`REPEAT`			
`ENDFOR`			
`DayNumber ← DayNumber + 1`			
`Error ← TRUE`			

[6]

(b) **(i)** The following table contains statements written in pseudocode.

Give the most appropriate data type for the variable used in each statement.

Statement	Data type
`Revision ← 500`	
`FuelType ← 'P'`	
`MinValue ← -6.3`	
`ServiceDue ← FALSE`	
`ModelRef ← "W212DEC15"`	

[5]

(ii) The following table contains statements written in pseudocode.

Complete the table by evaluating each expression using the values from part **(b)(i)**.

If any expression is invalid, write “ERROR” in the **Evaluates to** column.

For the built-in functions list, refer to the **Appendix** on page 16.

Expression	Evaluates to
`"Month: " & MID(ModelRef, 5, 3)`	
`INT(MinValue * 2)`	
`ASC(Revision)`	
`Revision > 500`	
`ServiceDue = TRUE OR FuelType = 'P'`	

[5]

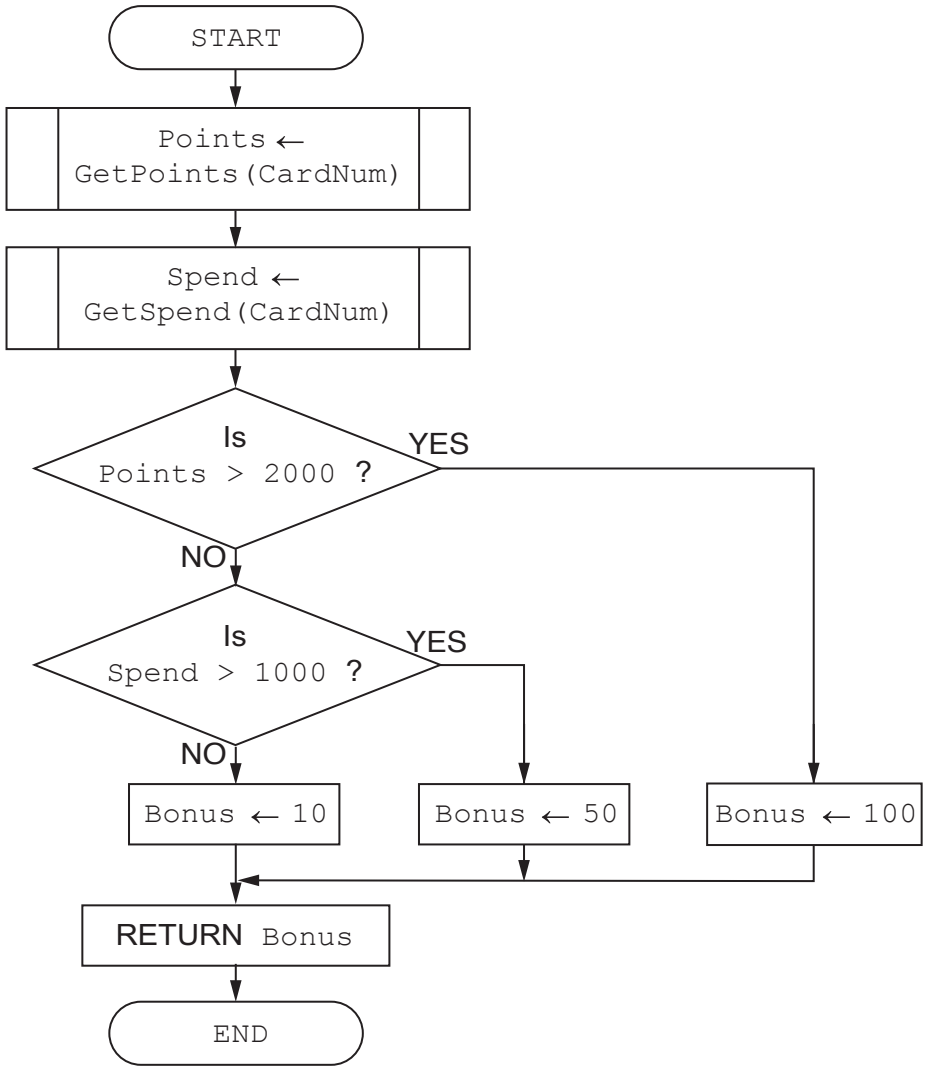
[Turn over

2 Shop customers have a discount card with a unique card number. Customers collect points when they buy items. At the end of each year, customers are given bonus (extra) points related to the total amount they have spent during the year, and the number of points they have on their card.

The function `CalcBonus()` takes the card number as a parameter. It returns the bonus points for the customer. A flowchart for the function is shown.

The function uses the following variables and functions.

Identifier	Data type	Description
`CardNum`	`STRING`	A numeric string representing the unique card number
`Points`	`INTEGER`	The number of points collected
`Spend`	`REAL`	The total amount that customer has spent during the year
`Bonus`	`INTEGER`	The number of bonus points
`GetPoints()`	`FUNCTION`	Takes the card number as a parameter and returns the number of points already collected
`GetSpend()`	`FUNCTION`	Takes the card number as a parameter and returns the total amount that customer has spent during the year



(a) (i) Write **pseudocode** for the `CalcBonus()` function.

Your solution should follow the flowchart for the function as closely as possible.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[5]

(ii) The function `GetCardNumber()` prompts the user to input a card number until the number input is valid.

A valid card number has 16 characters. Each character is a numeric character (`'0'` to `'9'`).

Write **pseudocode** to complete the `GetCardNumber()` function.

You should refer to the function `IS_NUM()` in the **Appendix** on page 16.

```
FUNCTION GetCardNumber() RETURNS STRING
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDFUNCTION
```

[6]

(b) The function `CalcBonus()` is written in a high-level language.

(i) The function is tested using black-box testing and does not contain any syntax errors.

Name **and** describe **one** other type of error that black-box testing could find.

Name .....

Description .....

.....

.....

[2]

(ii) The function `CalcBonus()` is tested using white-box testing.

State **two** different pairs of values for `Spend` and `Points` that can be used to test different paths through the function. Justify your choices.

`Spend` ..... `Points` .....

Justification .....

.....

.....

`Spend` ..... `Points` .....

Justification .....

.....

.....

[4]

(c) Name **two** types of program maintenance **and** state the reason why each is needed.

Name .....

Reason .....

.....

.....

Name .....

Reason .....

.....

.....

[4]

3 An array contains 100 integer values. An algorithm will find the maximum and minimum values stored in the array.

(a) A programmer has started to write this program using a conditional loop.

Name a more appropriate loop structure for this task **and** justify your choice.

Name .....

Justification .....

.....

.....

[2]

(b) Outline the steps the program will need to follow to implement the algorithm.

Do **not** write pseudocode or program code.

.....

.....

.....

.....

.....

.....

.....

.....[3]

Question 4 begins on the next page.

4 Part of a program written in pseudocode is shown.

```
010 DECLARE ThisArray : ARRAY [1:100, 1:10] OF STRING
011 DECLARE LastElement : INTEGER
…
100 FUNCTION Update(NewData : STRING) RETURNS INTEGER
101    DECLARE ArrayIndex : INTEGER
102    DECLARE Found : BOOLEAN
103
104    ArrayIndex ← 1
105    Found ← FALSE
106
107    WHILE ArrayIndex <= LastElement AND Found = FALSE
108       IF ThisArray[ArrayIndex, 1] > NewData
109         THEN
110           Found ← TRUE
111         ELSE
112           ArrayIndex ← ArrayIndex + 1
113       ENDIF
114    ENDWHILE
115
116    IF Found = TRUE
117      THEN
118         CALL Insert(ArrayIndex, NewData)
119      ELSE
120         ArrayIndex ← 0
121    ENDIF
122
123    RETURN ArrayIndex
124 ENDFUNCTION
```

(a) (i) Examine the pseudocode **and** complete the following table.

	Answer
The name of a global identifier	
The name of a user-defined procedure	
The scope of `ArrayIndex`	
The number of dimensions of `ThisArray`	
The scope of `NewData`	

[5]

(ii) Describe in detail, the purpose of lines `107` to `114` in the `Update()` function.
Do **not** use pseudocode in your answer.

.....

.....

.....

.....

.....

.....[4]

(b) Line `118` of the function `Update()` calls the subroutine `Insert()`. A designer decides to convert `Insert()` from a procedure to a function. `Insert()` returns `TRUE` or `FALSE`.

A programmer must amend the function `Update()` as follows:

- If `Insert()` returns `FALSE` then the function `Update()` returns `-1`
- If `Insert()` returns `TRUE` then the function `Update()` returns the value as before.

Write **program code** to implement the **amended** `Update()` function.

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]

(c) The function `Update()` is an example of a module within a program.

Describe the mechanism that supports the transfer of values between modules.

.....

.....

.....

.....

.....[2]

(d) **(i)** `CharArray` is a 1D array of type `CHAR`. It contains 200 elements.

Write **program code** to change all the numeric characters (`'0'` to `'9'`) in `CharArray` to `'*'`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

.....

.....[3]

(ii) A programmer decides to declare `LastElement` as a constant instead of a variable.

Write a statement in **pseudocode** to declare `LastElement` as the value 200.

.....[1]

Question 5 begins on the next page.

5 The function `ReadFileLine()` returns a specific line from a text file.

The function takes two parameters:

Identifier	Data type	Description
`FileName`	`STRING`	The name of the text file
`FileLine`	`INTEGER`	The line number that is required

The following pseudocode gives an example of the use of the function.

```
FileData ← ReadFileLine(FileName, FileLine)
```

The function `ReadFileLine()` will:

- open the file, `FileName`
- read each line from the file until line `FileLine` is found or the end of the file is reached
- if the line exists, return the string from this line; otherwise return the string `"****"`

Write **pseudocode** for the `ReadFileLine()` function.

....[10]

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 string `"BCD"`

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

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

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

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

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

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

`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`

`IS_NUM(ThisString : STRING) RETURNS BOOLEAN`
returns the value `TRUE` if `ThisString` contains only numeric characters (`'0'` to `'9'`).

Example: `IS_NUM("1234X67")` returns `FALSE`

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`