Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME	
CENTRE NUMBER	
CANDIDATE NUMBER	

2720504716

COMPUTER SCIENCE **9608/22**

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

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	Selection	Repetition (Iteration)	Assignment
`Index ← Index + 5`			
`FOR Count ← 1 TO 100`			
`TempValue[Index] ← ReadValue(SensorID)`			
`IF Index < 30`			
`UNTIL DayNumber > 7`			
`OTHERWISE OUTPUT "ERROR"`			

[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 ← 'B'`	
`MaxValue ← 13.3`	
`ArrayFull ← TRUE`	
`Activity ← "Design"`	
`NumberOfEdits ← 270`	

[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
`MID(Activity, 3, 4) & "ature"`	
`INT(MaxValue * 2)`	
`ArrayFull AND NumberOfEdits < 300`	
`ASC(Revision + 1)`	
`Activity = "Testing" OR Revision = 'A'`	

[5]

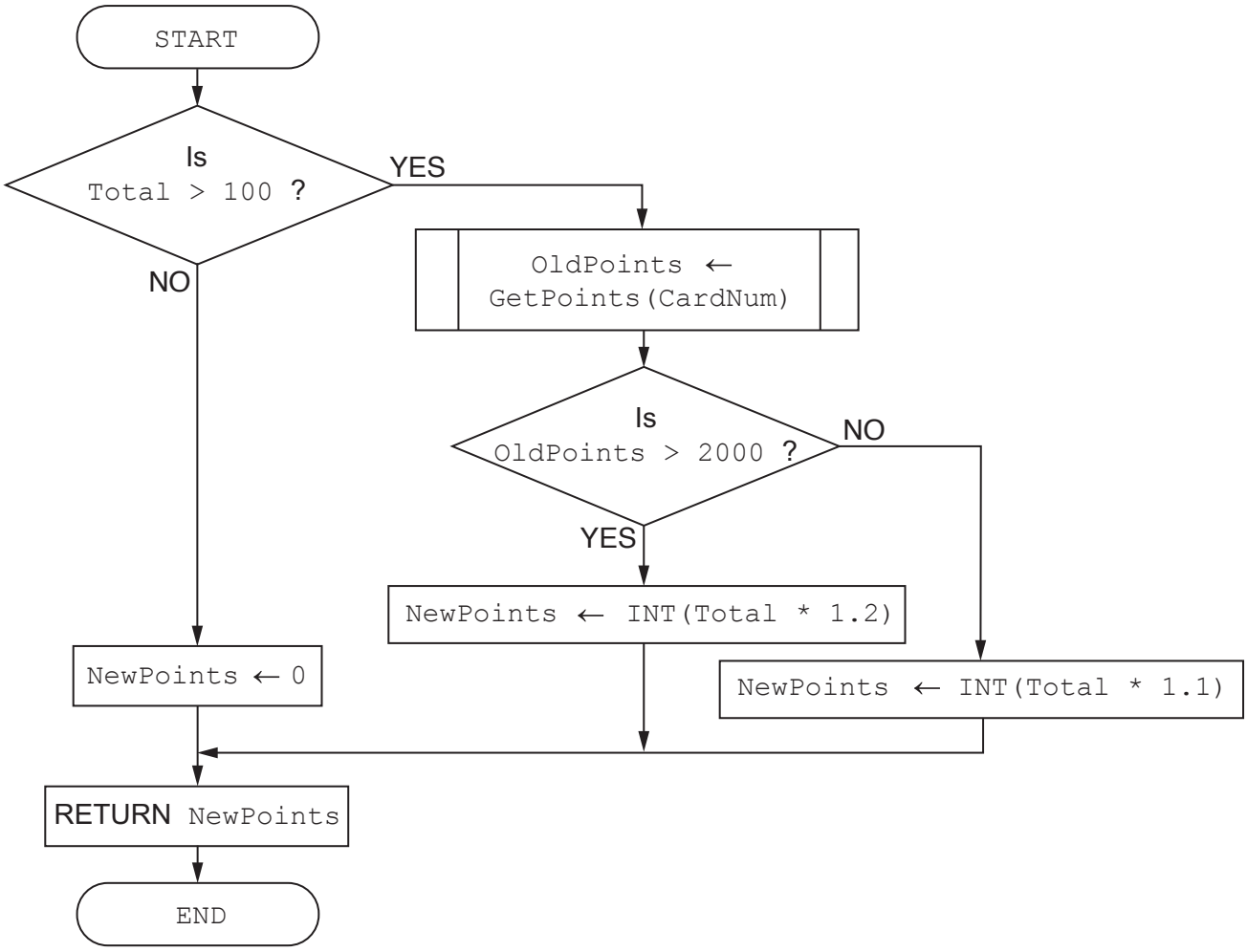
2 Shop customers have a discount card with a unique card number. Customers collect points each time they buy items. The number of points they collect depends on:

- the total amount they spend
- the number of points already collected.

The function `CalcPoints()` takes the card number and the total amount spent as parameters. It returns the number of new points collected. 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
`OldPoints`	`INTEGER`	The number of points already collected
`NewPoints`	`INTEGER`	The number of new points collected
`Total`	`REAL`	The amount spent
`GetPoints()`	`FUNCTION`	Takes the card number as a parameter and returns the number of points already collected
`INT()`	`FUNCTION`	Refer to the **Appendix** on page 16



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

Your solution should follow the flowchart as closely as possible.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[7]

(ii) The value of the total amount spent is calculated by an Electronic Point Of Sale (EPOS) system. This system does not have the prices of all items. For these items, a valid total amount has to be entered manually.

A function, `GetTotal()`, prompts the user to input this value.

If the user enters a valid value greater than 0 and less than 10000, the function returns the value. The function prompts the user to re-enter the value each time the user enters an invalid value.

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

```
FUNCTION GetTotal() RETURNS REAL
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

```
ENDFUNCTION
```

[5]

(b) The function `CalcPoints()` is written in a high-level language. It has been checked and it does not contain any syntax or logic errors.

(i) Name **and** describe **one** other type of error that the high-level language code could contain.

Name .....

Description .....

.....

.....

[2]

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

State **two** different values of `Total` that could be used to test different paths through the algorithm. Justify your choices.

Value .....

Justification .....

.....

.....

Value .....

Justification .....

.....

.....

[4]

3 **(a)** Programming is sometimes referred to as a **transferable skill**.

You are asked to work on a program written in a language you are not familiar with.

Explain how **transferable skills** would help you work on the program.

.....

.....

.....

.....[2]

(b) Stepwise refinement is often used in the development of an algorithm.

Describe **stepwise refinement**.

.....

.....

.....

.....[2]

(c) A program needs to search through 1000 elements of an unsorted array to find a given value.

The program will output:

- either the position in the array of the value
- or the message “Not Found”

Outline the steps the program needs to follow.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

Question 4 begins on the next page.

4 Part of a program written in pseudocode is shown.

```
010 DECLARE NextArrayElement : INTEGER
...
100 FUNCTION ScanFile(SearchString: STRING) RETURNS INTEGER
101
102    DECLARE FileData : STRING
103    DECLARE FileLine : INTEGER
104
105    NextArrayElement ← 1
106    FileLine ← 1
107    FileData ← ReadFileLine("DataFile.txt", FileLine)
108
109    WHILE FileData <> ""
110        IF LEFT(FileData, 7) = SearchString
111          THEN
112            ResultArray[NextArrayElement] ← FileData
113            NextArrayElement ← NextArrayElement + 1
114        ENDIF
115        FileLine ← FileLine + 1
116        FileData ← ReadFileLine("DataFile.txt", FileLine)
117    ENDWHILE
118
119    CALL ScanCompleted()
120    RETURN FileLine
121
122 ENDFUNCTION
```

(a) (i) Examine the pseudocode. Complete the following table.

	Answer
The identifier name of a local variable	
The identifier name of a user-defined procedure	
The identifier name of a user-defined function	
The number of dimensions of `ResultArray`	
The scope of `FileData`	

[5]

(ii) Describe in detail the purpose of lines `109` to `117` in the `ScanFile()` function. Do **not** use pseudocode in your answer.

.....

.....

.....

.....

.....

.....[4]

(b) The function `ScanFile()` is to be amended so that the first 7 characters of `FileData` are **not** written to the array.

Write **program code** to implement this **amended** `ScanFile()` 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) **(i)** State how structured programming languages support the implementation of sub-tasks.

.....[1]

(ii) State a benefit of using sub-tasks.

.....

.....[1]

(d) `ResultArray` is a 1D array of type `STRING`. It contains 100 elements.

Write **program code** to declare `ResultArray` and set all elements to the value `"NO DATA"`.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....[3]

Question 5 begins on the next page.

5 The procedure `LineNumber()` will:

- read data from a text file
- modify each line by adding a line number, and the string (`": "`)
- output each modified line.

For example, when the procedure reads `MyFile.txt`, the output is:

```
10: <First line of MyFile.txt>
15: <Second line of MyFile.txt>
20: <Third line of MyFile.txt>
…
350: <Last line of MyFile.txt>
```

The procedure takes three parameters:

Identifier	Data type	Description
FileName	`STRING`	The name of the text file
StartNumber	`INTEGER`	The first line number to be added
StepNumber	`INTEGER`	The line number increment

In this example, the procedure call would be:

```
CALL LineNumber("MyFile.txt", 10, 5)
```

After every 20 lines, the program outputs a message asking whether the user wants to continue.

The program ends when the user enters an `'N'` or the end of file is reached.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....[11]

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"`

`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 string `"ABC"`

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

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

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value.

Example: If `x` has the value `87.5` then `NUM_TO_STRING(x)` will return `"87.5"`
Note: This function will also work if `x` is of type integer.

`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 character `ThisChar`

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

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`