Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

CANDIDATE NAME

CENTRE NUMBER | | | | | |

CANDIDATE NUMBER | | | | |

4324571957

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2017**

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 **13** printed pages and **3** blank pages.

DC (NF) 129956/4

CAMBRIDGE International Examinations

[Turn over

1 **(a)** Simple algorithms usually consist of input, process and output.

The statements in the following table are in a generic programming language.

Complete the table by placing ticks in the relevant boxes.

Item	Statement	Input	Process	Output
1	`String1 = "Hello World"`			
2	`DISPLAY RIGHT(String1, 5)`			
3	`READFILE (MyFile, String2)`			
4	`WRITEFILE (MyFile, "Data is " & String2)`			

[6]

(b) **(i)** Complete the following two sentences.

A suitable operand type for an arithmetic operator is .....

A suitable operand type for a logical operator is .....

[2]

(ii) The following table shows the values of three variables.

Variable	Value
`FlagA`	`TRUE`
`FlagB`	`FALSE`
`FlagC`	`TRUE`

Evaluate these expressions.

Expression	Evaluates to
`(FlagA AND FlagB) OR FlagC`	
`FlagA AND (FlagB OR FlagC)`	
`(NOT FlagA) OR (NOT FlagC)`	

[3]

(c) The loop construct (also known as repetition or iteration) appears in many algorithms.

Use **pseudocode** to write a post-condition loop to output all the odd numbers between 100 and 200.

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

2 A multi-user computer system maintains a text file containing the ID and preferred name for each user.
User IDs are unique. Preferred names may be repeated.

(a) Stepwise refinement is to be applied to the following three steps.

After a user logs in, a welcome message is produced as follows:

1. Search for the user ID in the file.
2. Read the preferred name from the file.
3. Output the welcome message.

Describe the goal of **stepwise refinement**.

.....

.....

.....

..... [2]

(b) An initial identifier table is created as part of the stepwise refinement. A section of the table is shown. Complete this table.

Identifier	Data type	Description
`SearchUserID`		Stores the user ID entered
`FileUserID`		....
`FilePreferredName`		....
`IDFoundFlag`		....

[5]

(c) The file may not contain an entry for every user. If the user ID is not found, then the name "Guest" should be used as the preferred name.

Use **structured English** to write a more detailed algorithm for the welcome message process described in **part (a)**.

Use the identifiers given in the previous table.

Assume that the file, `UserNames.txt`, is organised as a simple text file. The user ID and preferred name are held together in unencrypted form, one entry per line.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

3 A string conversion function, `ExCamel`, needs to be written.

This function forms a return string, `OutString`, from a given string, `InString`, by:

1 separating the original words (a word is assumed to start with a capital letter)
2 converting all characters to lower case.

The following shows a pair of example values for the string values `InString` and `OutString`.

```
InString : "MyUserInput"
OutString : "my user input"
```

You may assume that `InString` always starts with a capital letter.

The following is a first attempt at writing the pseudocode for this function.

Complete the **pseudocode** using appropriate built-in functions.

For the built-in functions list, refer to the **Appendix** on page 13.

```
FUNCTION ExCamel (.........................) RETURNS .............
   DECLARE NextChar : ......................................
   DECLARE ................................................... : STRING
   DECLARE n: INTEGER
   ..................................... // initialise the return string
   // loop through InString to produce OutString
   FOR n ← 1 TO ............................ // from first to last
      NextChar ← ........................... // get next character
      IF ................................... // check if upper case
         THEN
            IF n > 1                        // if not first character
               THEN
                  ........................... // add space to OutString
            ENDIF
            ................................. // make NextChar lower case
      ENDIF
      ..................................... // add NextChar to OutString
   ENDFOR
   ..................................... // return value
ENDFUNCTION
```

[11]

4 **(a)** High-level programming languages have many features that support the modular approach. One such feature is the use of parameters.

State **two** other features.

1 .....

.....

2 .....

.....

[2]

(b) Consider the following pseudocode.

```
PROCEDURE MyProc(x)
   x ← x + 1
ENDPROCEDURE

Intermediate lines of pseudocode not shown

x ← 4
CALL MyProc(x)
OUTPUT(x)
```

Parameter `x` is used to pass data to procedure `MyProc`.
There are two parameter passing methods that could be used.

Complete the following table for each of the two methods.

Name of parameter passing method	Value output	Explanation
....	.....	....
....	.....	....

[6]

5 A multi-user computer system records user login data. Each time a user successfully logs into the system, it records the following data.

Data item	Example data
User ID	`"Jim27"`
Port ID	`"3456"`
Time and date	`"08:30 Jun 01 2015"`

The data items are concatenated (joined) using a separator character to form a single string. Each string represents one log entry.

(a) (i) Suggest a suitable separator character. Give the reason for your choice.

Character .....

Reason .....

.....

[2]

(ii) The concatenated strings are stored in an array, `LogArray`, which may contain up to 20 log entries.

Use **pseudocode** to declare `LogArray`.

..... [2]

(b) A procedure, `LogEvents`, is required to add the log entry data from `LogArray` to the existing text file, `LoginFile.txt`.

Unused array elements are assigned the value `"****"`. These can occur anywhere in the array and should not be written to the file.

Write **program code** for the procedure `LogEvents`.

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]

6 A computerised vehicle licensing system stores details about vehicles and their registration marks (number plates or license plates).

To be valid, a vehicle registration must comply with the following rules:

- It must be between six and nine characters long.
- Characters 1 to 3 are upper case alphabetic characters.
- Characters 4 to 5 are numeric characters.
- Remaining characters are upper case alphabetic.

A function, `ValidateRegistration` is needed to check that a given registration mark follows these rules. This function takes a string, `Registration` as a parameter and returns a Boolean value:

- `TRUE` if it is a valid registration
- `FALSE` otherwise.

(a) Write **program code** to implement the new function, `ValidateRegistration`.

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [9]

(b) The function is to be tested.

Give a valid string that could be used to check that the function returns TRUE under the correct conditions.

String1: .....

Modify your valid String1 to test each rule separately.

Explain your choice in each case.

String2: .....

Explanation: .....

.....

.....

String3: .....

Explanation: .....

.....

.....

String4: .....

Explanation: .....

.....

.....

String5: .....

Explanation: .....

.....

.....

[5]

Appendix

Built-in functions (pseudocode)

In each function, if the function call is not properly formed, the function returns an error.

`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 string `ThisString`.
Example: **`LENGTH("Happy Days")`** returns **`10`**

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`

returns string of length `y` starting at position `x` from `ThisString`.
Example: **`MID("ABCDEFGH", 2, 3)`** returns string **`"BCD"`**

`LCASE(ThisChar : CHAR) RETURNS CHAR`

returns the character value representing the lower case equivalent of `ThisChar`.
If `ThisChar` is not an upper case alphabetic character then it is returned unchanged.
Example: **`LCASE('W')`** returns **`'w'`**

`UCASE(ThisChar : CHAR) RETURNS CHAR`

returns the character value representing the upper case equivalent of `ThisChar`.
If `ThisChar` is not a lower case alphabetic character then it is returned unchanged.
Example: **`UCASE('h')`** returns **`'H'`**

`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`**

`DIV(ThisNum : INTEGER, ThisDiv : INTEGER) RETURNS INTEGER`

returns the integer value representing the whole number part of the result when `ThisNum` is divided by `ThisDiv`.
Example: **`DIV(10,3)`** returns **`3`**

Operators (pseudocode)

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

BLANK PAGE

BLANK PAGE

BLANK PAGE