Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/21**

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

PRE-RELEASE MATERIAL

No Additional Materials are required.

This material should be given to the relevant teachers and candidates as soon as it has been received at the Centre.

READ THESE INSTRUCTIONS FIRST

Candidates should use this material in preparation for the examination. Candidates should attempt the practical programming tasks using their chosen high-level, procedural programming language.

DC (ST) 129909/4

CAMBRIDGE International Examinations

This material is intended to be read by teachers and candidates prior to the June 2017 examination for 9608 Paper 2.

Reminders

The syllabus states:

- there will be questions on the examination paper which do not relate to this pre-release material.
- you must choose a high-level programming language from this list:
  - Visual Basic (console mode)
  - Python
  - Pascal / Delphi (console mode)

Note: A mark of **zero** will be awarded if a programming language other than those listed is used.

Questions on the examination paper may ask the candidate to write:

- structured English
- pseudocode
- program code

A program flowchart should be considered as an alternative to pseudocode for the documenting of an algorithm design.

Candidates should be confident with:

- the presentation of an algorithm using either a program flowchart or pseudocode
- the production of a program flowchart from given pseudocode or the reverse.

There is an **Appendix** at the end of this document. Some tasks refer you to this information.

There will also be a similar appendix at the end of the question paper.

Declaration of variables

The syllabus document shows the syntax expected for a declaration statement in pseudocode.

```
DECLARE <identifier> : <data type>
```

If Python is the chosen language, each variable's identifier (name) and its intended data type must be documented using a comment statement.

Structured English – Variables

An algorithm in pseudocode uses variables, which should be declared. An algorithm in structured English does not always use variables. In this case, the candidate needs to use the information given in the question to complete an identifier table. The table needs to contain an identifier, data type and description for each variable.

TASK 1 – Stages of a Simple Algorithm

(a) Fundamental algorithmic stages: Input, Process and Output.

TASK 1.1

Describe a range of activities in terms of these three stages.

Draw examples from different areas such as:

- Manufacturing
- College / School
- Data processing systems

Key focus:
Input, process, output

TASK 1.2

Give examples of programming statements that illustrate each of the three stages.

Compare similar statements in pseudocode and high-level languages.

(b) Appreciate the use of logic statements within an algorithm.

TASK 1.3

Given the following pseudocode assignments:

Key focus:
Logic statements

```
FlagA ← TRUE
FlagB ← FALSE
FlagC ← TRUE
MyNum ← 27
```

Evaluate the following expressions:

Expression	Evaluates to
`FlagA AND FlagB`	
`FlagB AND FlagC`	
`FlagB OR FlagC`	
`FlagA AND (FlagB OR FlagC)`	
`FlagB AND (NOT FlagB)`	
`(MyNum > 27) AND FlagC`	
`(MyNum >= 27) AND (FlagC = FALSE)`	

[Turn over

TASK 2 – Programming Basics: Loops

This is the pseudocode for a simple count-controlled loop:

Key focus:
A simple count-controlled loop

```
FOR Count ← 1 TO 100
   OUTPUT Count
ENDFOR
```

TASK 2.1

Re-write this pseudocode to provide the same functionality using:

1. A post-condition loop
2. A pre-condition loop

Key focus:
Different types of loops

TASK 2.2

Modify the solutions from TASK 2.1 to:

1. Prompt for the input of the start and end value.
2. Output all the numbers between the start and end values that are exactly divisible by 3.

For the built-in functions list, refer to the **Appendix** on page 7.

Key focus:
More interesting loops...

TASK 3 – Program design and coding

A string exists in CamelCase format. For example: "ThisIsACamelCaseString".

A procedure is required that will:

- prompt for the original string
- separate each word of the string
- store each word in a separate array element
- fill unused array elements with a rogue string such as "(Empty)".

After processing the preceding example, the array contents will look like this:

This
Is
A
Camel
Case
String
(Empty)
(Empty)
(Empty)
(Empty)

Key focus:
Pseudocode array declaration

TASK 3.1

Use pseudocode to declare the array that can be used to store the separate words of the original string. You can assume that the original string will contain no more than 10 separate words.

TASK 3.2

Express the requirements using structured English.

Key focus:
Algorithm using structured English

TASK 3.3

Write **pseudocode** for this design.
For the built-in functions list, refer to the **Appendix** on page 7.

TASK 3.4

Write **program code** for this design.

Key focus:
Pseudocode and program code

TASK 3.5

Amend your program code to print out the contents of the array in a format that is easy to understand.

TASK 3.6

Design suitable test data to test the code thoroughly. Justify your choice.

Key focus:
Test data

[Turn over

TASK 4 – File handling

The computer system at a sports club maintains a log of attendances of each member. Each time a member visits the club, an entry is added to a log file as follows:

- Membership number (6 characters, for example `"123456"`)
- Date of visit (8 characters, for example `"28/07/15"`)

The system concatenates these data items and adds them as a single line to the file.

Key focus:
Create a text file and add data

TASK 4.1

Write pseudocode to create a text file containing log data for a number of attendances by sports club members. The user will input the two strings described above and these will be concatenated and added to the file.

The process will repeat until a rogue value is input.

TASK 4.2

Write pseudocode to append a new data line to the existing file.

Key focus:
Append data to an existing file

TASK 4.3

Write pseudocode to output a list of visits made by a member. Prompt the user to input the membership number and output the date of each visit by that member. If the membership number is not found, output a suitable message.

Key focus:
Search for data in a file

TASK 4.4

Write program code for TASK 4.1 to TASK 4.3.

TASK 4.5

Extend the program from TASK 4.4. The program will present a menu to allow the user to repeatedly add a new visit (4.2) or print the visits (4.3). Use a suitable input value to terminate the program.

Key focus:
Implement a menu interface

Appendix

Built-in functions (pseudocode)

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

`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"`**

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

`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