Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2020**

PRE-RELEASE MATERIAL

8055895233

No additional materials are needed.

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

INSTRUCTIONS

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

This document has **8** pages. Blank pages are indicated.

DC (PQ) 199034

Teachers and candidates should read this material prior to the June 2020 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 documenting 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 and vice versa.

Some tasks may need one or more of the built-in function or operators listed in the **Appendix** at the end of this document. 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 – Structure charts

Describe a processing activity that can be represented by one main task with two or more sub-tasks. The activity can relate to any scenario, but should include aspects of selection and iteration.

Activity examples may be taken from different areas, such as:

- school or college
- factory or workplace
- clubs or hobbies.

TASK 1.1

Consider how a problem is decomposed by splitting it into smaller parts.

Discuss the advantages of this approach.

TASK 1.2

Design a modular program to implement the activity described in TASK 1.

Produce a structure chart to represent the modular structure of the solution.

The structure chart should address:

- the sequence of module execution
- any module selection or iteration
- the parameters that are passed between the modules.

TASK 1.3

For each module, decide whether the solution should be implemented as a procedure or a function.

Justify your choices.

Produce **pseudocode** headers for each module.

TASK 2 – Algorithms, arrays and pseudocode

Declare an array to store multiple pieces of data for your activity in TASK 1. For example, a 1D array of `STRING` could store the names of students in a class.

TASK 2.1

Design an algorithm to search for a specific value in the array and output the array index where the value is found.

Consider the differences between algorithms that search for a single rather than multiple instances of the value.

Document the algorithm using:

- structured English
- a program flowchart
- pseudocode.

TASK 2.2

Design an algorithm to manipulate data in the array, for example by sorting.

Document the algorithm as in TASK 2.1.

TASK 3 – Programs containing several components

A library maintains a list of books. The list is saved in a text file, where each line of the file represents one book.

TASK 3.1

Consider the information that should be included in the text file other than the title and the author.

TASK 3.2

Consider that this is a text file, which means that all information will be saved in `STRING` format.

Consider the implications of storing numeric information, such as the number of copies of each book.

Define the format of each line of the file so that each piece of information may be easily extracted.

TASK 3.3

Design a program in **pseudocode** that has a menu-driven interface and will perform the following tasks:

1. Add a new book to the text file. Include validation of the different pieces of information as appropriate.

2. Search for books written by a given author. Output the title of any books found, or a suitable message if no books by the given author are found.

3. End the program.

TASK 3.4 – Writing program code

Convert your pseudocode into **program code**.

TASK 3.5 – Testing

Consider how the program produced in TASK 3.4 may be tested.

TASK 4 – Algorithm modification

Additional information needs to be saved for each book, such as a publication date.

An additional task is needed to create a new file from the original file. The task will prompt the user to input the additional information for each book.

Consider possible validation of the additional information.

Write **program code** for the additional task.

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", 4)` returns `"EFGH"`

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

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

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value.
Note: This function will also work if `x` is of type `INTEGER`

Example: `NUM_TO_STRING(87.5)` returns `"87.5"`

`STRING_TO_NUM(x : STRING) RETURNS REAL`
returns a numeric representation of a string.
Note: This function will also work if `x` is of type `CHAR`

Example: `STRING_TO_NUM ("23.45")` returns `23.45`

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`

BLANK PAGE

BLANK PAGE