Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2021**

PRE-RELEASE MATERIAL

9925540472

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. Any blank pages are indicated.

DC (SLM) 220517

Candidates and teachers should read this material prior to the November 2021 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 functions 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 – Structured programming

TASK 1.1

Describe a process using an algorithm written using structured English.

Use examples from:

- sports or leisure clubs
- college or school
- hotels.

TASK 1.2

Split the process into sub-tasks and consider the advantages of this approach.

TASK 1.3

Produce a structure chart for the design of this modular program.

TASK 1.4

Convert each of the sub-tasks into a program flowchart.

TASK 1.5

Convert the program flowcharts into pseudocode algorithms and then into a high-level language program.

TASK 1.6

Research the different types of errors that can occur at different stages of the program development cycle.

TASK 1.7

Consider the advantages of modular programming using both built-in and user-defined functions.

TASK 1.8

Consider ways of testing the complete program both with and without knowledge of the underlying program code.

Consider ways of testing the main program before all the user-defined functions have been completed.

TASK 2 – Good programming practice

TASK 2.1

Discuss what is meant by **good programming practice**.

TASK 2.2

Design and write program code to declare, initialise and output the contents of a 1D array.

Implement different initialisation methods such as:

- Fill the complete array with a single value.
- Fill the array with an incrementing or decrementing sequence of values.
- Fill the array with values obtained using a programmed function, for example, a Fibonacci sequence.

Add a simple menu interface to allow the user to repeatedly select any of the previous stages and to output the array elements.

Use good programming practice throughout.

TASK 2.3

Swap your program with one of your peers.

Amend this program so that it processes the elements in the array. An example of a process is to add up all the elements in a range and find the average value.

TASK 2.4

Modify the program to work with a 2D array.

TASK 2.5

Describe in detail the purpose of a section of code you have not written yourself.

TASK 3 – File handling

The computer system at a sports club maintains a log of visits by each member.

Each time a member visits the club, an entry is added to a file as follows:

- Membership number (6 characters, for example "123456")
- Date of visit (8 characters, for example "28/09/20")

The system concatenates these data items and stores them as a single line in the file.

TASK 3.1

Write pseudocode to create a text file containing log data for several visits by the sports club members.

The user will input the two strings described above and these will be concatenated and added to the file.

TASK 3.2

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

The process will repeat until a rogue value is input.

TASK 3.3

Write pseudocode to output a list of visits made by a sports club 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.

TASK 3.4

Write program code for TASK 3.1 to TASK 3.3.

TASK 3.5

Extend the program from TASK 3.4 to present a menu to allow the user to repeatedly add a new visit (TASK 3.2) or print the visits (TASK 3.3).

Use a suitable input value to terminate the program.

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", 3)` returns `"FGH"`

`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`

`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`

`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