Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9608/21**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2020**

PRE-RELEASE MATERIAL

5691260284

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.

DC (LK) 188574/2

[Turn over

Teachers and candidates should read this material prior to the November 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 – Algorithms, arrays and pseudocode

The following four 1D arrays can store different pieces of data about stock items in a shop:

Array identifier	Data type
`ItemCode`	`STRING`
`ItemDescription`	`STRING`
`Price`	`REAL`
`NumberInStock`	`INTEGER`

TASK 1.1

Design an algorithm to search for a specific value in `ItemDescription` and, if found, to output the array index where the value is found. Output a suitable message if the value is not found.

Document the algorithm using:

- structured English
- a program flowchart
- pseudocode.

TASK 1.2

Consider the difference between algorithms that search for a single or multiple instance of the value.

TASK 1.3

Extend the algorithm from Task 1.1 to output the corresponding values from the other arrays.

TASK 1.4

Write **program code** to produce a report displaying all the information stored about each item for which the number in stock is below a given level.

[Turn over

TASK 2 – Programs containing several components

The stock data described in Task 1 are now to be stored in a text file.

Each line of the file will correspond to one stock item.

TASK 2.1

Define a format in which each line of the text file can store the different pieces of data about one stock item.

Consider whether there is a requirement for data type conversion.

TASK 2.2

Design an algorithm to input the four pieces of data about a stock item, form a string according to your format design, and write the string to the text file.

First draw a program flowchart, then write the equivalent pseudocode.

TASK 2.3

Write **program code** to implement your algorithm.

TASK 2.4

Consider the different modes when opening a file.

Discuss the difference between creating a file and amending the contents.

Extend the program to include a menu-driven interface that will perform the following tasks:

A. Add a new stock item to the text file. Include validation of the different pieces of information as appropriate. For example, item code data may be a fixed format.

B. Search for a stock item with a specific item code. Output the other pieces of data together with suitable supporting text.

C. Search for all stock items with a specific item description, with output as for task B.

D. Output a list of all stock items with a price greater than a given amount.

TASK 3 – Testing

TASK 3.1

You need to design tests to prove that the program works as expected.

Create a table for a test plan, with columns for:

- data item tested
- type of test data (to explain why you choose the test data value)
- test data value
- expected output
- actual output.

Complete the test plan.

TASK 3.2

Discuss different testing methods such as black-box, white-box and stub testing.

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 string `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", 4)` returns string `"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