Cambridge International AS & A Level

Cambridge Assessment International Education
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2019**

PRE-RELEASE MATERIAL

6581430813

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.

Teachers and candidates should read this material prior to the June 2019 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 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 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 – Arrays

Introduction

Candidates should be able to write programs to process array data both in pseudocode and in their chosen programming language. It is suggested that each task is planned using pseudocode before writing it in program code.

TASK 1.1

A 1D array of `STRING` data type will be used to store the name of each student in a class together with their email address as follows:

```
<StudentName>'#'<EmailAddress>
```

An example string with this format would be:

```
"Eric Smythe#eric@email.com"
```

Write **program code** to:

1. declare the array
2. prompt and input name and email address
3. form the string as shown
4. write the string to the next array element
5. repeat from step 2 for all students in the class
6. output each element of the array in a suitable format, together with explanatory text such as column headings

TASK 1.2

Consider what happens when a student leaves the class and their data item is deleted from the array.

Decide on a way of identifying unused array elements and only output elements that contain student details. Modify your program to include this.

TASK 1.3

Extend your program so that after assigning values to the array, the program will prompt the user to input a name, and then search the array to find that name and output the corresponding email address.

TASK 1.4

Modify your program so that it will:

- prompt the user to input part, or the whole, of a name
- search the whole array to find the search term within the `<StudentName>` string
- for each array element in which the search term is found within the `<StudentName>` string, output the element in a suitable format.

TASK 1.5

Convert your design to use a 2D array and add additional pieces of information for each student.

For example:

Array element	Information	Example data
`MyArray[1,1]`	Student Name	`"Tim Smith"`
`MyArray[1,2]`	Email Address	`"TimSmith1099@email.com"`
`MyArray[1,3]`	Date of Birth	`"15/05/2001"`
`MyArray[1,4]`	Student ID	`"C3452-B"`

TASK 1.6

Modify your program to work with the new structure.

TASK 2 – Files

Introduction

Candidates should be able to write programs to process text file data both in pseudocode and their chosen programming language. It is suggested that each task is planned using pseudocode before writing it in program code.

TASK 2.1

Define a structure for a text file that could be used to store information about each student as a string. Each line of the file will contain a single string.

Store at least two pieces of information. For example, you could store each student's ID together with their email address as follows:

```
<StudentID>'#'<EmailAddress>
```

Define a fixed format for the Student ID, for example, two letters followed by four numbers.

An example string with this format would be:

```
"AB1234#Jim99@email.com"
```

Write **program code** to:

1. open a new text file
2. prompt and input a Student ID and email address (validate if required)
3. form the string as shown
4. write the string to the file
5. repeat from step 2 for all students in the class
6. close the file

Check the contents of the file using a text editor.

TASK 2.2

Write a second program to search the file for a given Student ID and output the email address if the ID was found, or a suitable message if the ID was not found.

TASK 2.3

Modify the search code to also perform a substring match on the Student ID. For example, search for all the Student IDs that begin with "AB".

TASK 2.4

Modify the program to allow the details of additional students to be appended to the file.

TASK 2.5

Modify the file structure to store different pieces of information on different lines of the file.

For example:

File line	Information	Example data
1	Student ID	"AB1234"
2	Email Address	"TimSmith1099@email.com"
3	Home Address	"1020 Red Canyon Road, Porthcawl"
4	Tutor	"Kim Lee"

TASK 2.6

Modify your program to work with the new structure.

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.

Example: If `x` has the value `87.5` then `NUM_TO_STRING(x)` returns `"87.5"`
Note: This function will also work if `x` is of type `INTEGER`

`STRING_TO_NUM(x : STRING) RETURNS REAL`
returns a numeric representation of a string.

Example: If `x` has the value `"23.45"` then `STRING_TO_NUM(x)` returns `23.45`
Note: This function will also work if `x` is of type `CHAR`

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