Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2021**

PRE-RELEASE MATERIAL

3076624995

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 (PQ) 211788

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

Introduction

Candidates should be able to write programs to process array data in pseudocode and in their chosen programming language. Each task should be planned using pseudocode before writing it in program code.

TASK 1.1

Design a way to store multiple pieces of data about a student in a single string. For example, you could store each student's name, email address and date of birth as follows:

```
<Student Name>'*'<Email address>'*'<Date of birth>
```

Example: "Sam Arnold*SamArnold137@email.com*25Sep2005"

Use a 1D array of type `STRING` to store data about each student in the class. Each element of the array will store data for one student.

Write **program code** to:

1. declare the array
2. prompt and input for student name, email address and date of birth
3. form the string as shown
4. assign the string to the next available array index
5. repeat from step 2 for all members of the class
6. output each element of the array in a suitable format, together with explanatory text such as column headers.

TASK 1.2

Consider what happens when a student's details are deleted from the array because the student has left the class.

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

TASK 1.3

Extend your program to:

- populate the array as in **Task 1.1**
- prompt the user to input a student name
- search the array to find that name and output the corresponding email address.

TASK 1.4

Extend Task 1.3 to output a list of all students who have a birthday in a given month.

TASK 1.5

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

For example:

Array element	Information	Example data
`MyArray[1,1]`	Student name	`"Sam Arnold"`
`MyArray[1,2]`	Email address	`"SamArnold137@email.com"`
`MyArray[1,3]`	Date of birth	`"25 Sep 2005"`
`MyArray[1,4]`	Student ID	`"C3452-B"`
`MyArray[1,5]`	Tutor ID	`"CHL"`

TASK 1.6

Modify your program to work with the new structure and extend the searches to work with any piece of data.

TASK 2 – Files

Introduction

Candidates should be able to write programs to process text file data in pseudocode and in their chosen programming language.

Each task should be planned using pseudocode before writing program code.

TASK 2.1

Define a structure for a text file used to store multiple pieces of data about each class member as a single string. Each line of the file will store data for one student.

Store at least three pieces of data. For example, you could store each student's ID together with his or her email address and date of birth as follows:

```
<Student ID><Email address><Date of birth>
```

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

Define a fixed format for the date of birth, for example, `"DDMMYY"`

An example string using this formatting would be:

```
"SA1234SamArnold137@email.com250905"
```

Write a program to:

1. open a new text file
2. prompt for the ID, email address and date of birth
3. form the string as shown
4. write the string to the file
5. repeat from step 2 for all members of 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 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

Consider rules that could be applied to ensure the data entered is acceptable.

Modify your program to incorporate these.

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 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 `"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