Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/22**

Paper 2 Fundamental Problem-solving and Programming Skills **October/November 2018**

PRE-RELEASE MATERIAL

2985316412

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 November 2018 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

Write an algorithm, using structured English to describe a process.

Draw examples from different areas such as:

- retail or banking
- sports or leisure clubs
- college or school
- hotels

Key focus:

Problem decomposition

TASK 1.2

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

TASK 1.3

Convert each sub-task into a program flowchart. Refer to the example program flowchart on page 4.

TASK 1.4

Convert each program flowchart into a pseudocode algorithm and then into a high-level language program.

TASK 1.5

Consider the different types of error that developers can identify in the program at different stages of its development cycle.

TASK 1.6

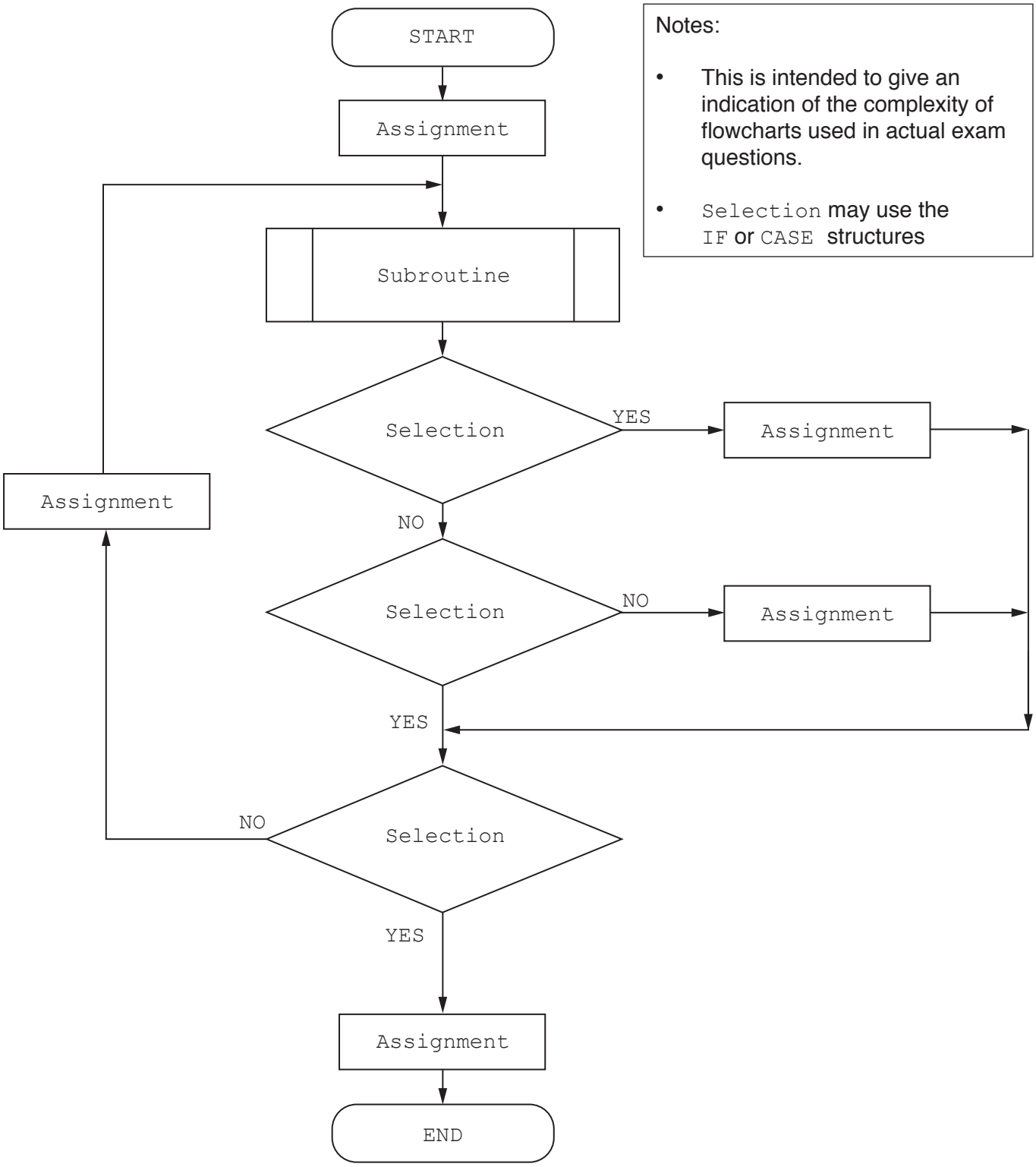
Consider the advantages of modular programming using both built-in and user-defined functions.

TASK 1.7

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 of the user-defined functions have been completed.

TASK 1 – Example program flowchart



TASK 2 – Good programming practice

TASK 2.1

The use of meaningful names for variables in a computer program is an example of good programming practice.

Discuss other examples of good programming practice.

Key focus:

Implement a program using good programming practice

TASK 2.2

Write a program 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 one of the previous initialisation methods
- output the contents of the array.

Use good programming practice throughout.

TASK 2.3

Swap your program with another student's program. Modify this program so that it processes the elements in the array.
For example, add up all the elements within 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.

Key focus:

Document unfamiliar code

TASK 3 – File handling

TASK 3.1

Write **program code** to create a multi-line text file. The program will prompt the user for the filename and the number of lines to be written. The program will write data to the file as follows:

This is line 1
This is line 2

This is line n

Key focus:

Text file handling in a high-level language

TASK 3.2

Open the file using a text editor and check that it contains the expected contents.

TASK 3.3

Write **program code** for a procedure to output data from the text file.

TASK 3.4

Write **program code** for a procedure to output a user-specified range of lines from the text file.

Check that your code outputs the lines of text in the correct format.

Key focus:

Dealing with the unexpected

TASK 3.5

List possible problems that your program could encounter when it attempts to output a user-defined range of lines from the text file.

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 `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", 3)` returns string `"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`

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