Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/23**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2016**

PRE-RELEASE MATERIAL

3445729518

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.

This material is intended to be read by teachers and candidates prior to the June 2016 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 (or the reverse)

There is an **Appendix** on the last page. Some tasks will refer you to this information.

There will also be a similar appendix on the last page of the question paper.

Some tasks require a candidate to write program code. These have been carefully chosen to encourage the candidate's programming skills to be at a standard in line with the question paper.

Please refer to section 7.2 of the current syllabus.

Declaration of variables

The syllabus document shows the syntax expected for a declaration statement in pseudocode.

```
DECLARE <identifier> : <data type>
```

It is appreciated that candidates who use Python as their chosen language will not be familiar with the concept of declaring all variables with their data type before they are used.

If Python is the chosen language, each variable's identifier (name) and its intended data type must be documented using a comment line.

The question will clarify this with a wording and answer layout such as:

> **(i)** Write **program code** for the new design.
>
> *Visual Basic and Pascal: You should include declaration statements for variables.*
> *Python: You should show a comment statement for each variable used with its data type.*
>
> Programming language .....
>
> .....
>
> .....

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 – String handling

Task 1.1

Evaluate the following expressions when `InputString ← "Computer Science"`

*For the built-in functions list, refer to the **Appendix** on the last page.*

- `LEFT(InputString, 4)`
- `MID(InputString, 4, 3)`
- `RIGHT(LEFT(InputString, 6), 3)`

Key focus:
Built-in functions

Task 1.2

Write program code to input a name in the format 'First name: Abdul, Second name: Sharif' and output:

- Abdul Sharif
- A Sharif
- Sharif, Abdul
- Mr A Sharif
- Dear Mr Sharif
- First name: Abdul, Second name: Sharif
- Abdul SHARIF

Task 2 – Program design and coding

The format of a user ID is the following sequence of characters:

- one upper case letter
- two lower case letters
- three numerals (digits)

Example: `"Hal123"`

The user ID must be validated.

The program will:

- take a user ID as input
- check the format
- output a message to show whether or not the format is correct

A program design is in structured English:

1. PROMPT for User ID
2. INPUT User ID
3. Check the format
4. IF User ID is in correct format, OUTPUT “Correct Format”
5. IF User ID is not in correct format, OUTPUT “Wrong Format”

Key focus:
Structured English and pseudocode

Task 2.1

Write the **pseudocode** equivalent of the structured English.

Task 2.2

Write **program code** for this design.

Key focus:
Pseudocode and program code

Extension tasks

Task 2.3

The requirement for the validation has changed. It now needs to accept any mix of upper and lower case letters as the first three characters.

Modify the program code from Task 2.2 to accommodate this change.

Task 2.4

Key focus:
User-defined functions

Redesign your program code as follows:

A user-defined function, `ValidateUserID`, is used to validate the user ID. It has the following function header:

```
FUNCTION ValidateUserID(ThisUserId : STRING) RETURNS BOOLEAN
```

Modify the program code from Task 2.2 to accommodate this change.

Task 3 – File handling

A text file is to be used to store the following membership information for a sports club:

- Name
- Member ID number (same format as user ID in Task 2.2)

Key focus:
Text files

Task 3.1

Write program code to create a text file containing membership data for a number of sports club members. The user will input data for several members. The program will save each data item on a new line of the file.

Task 3.2

Write program code to output the file contents as two columns, each with an appropriate heading.

Task 3.3

Write program code to search for a member name input by the user. If the name is found, output the corresponding ID number. If the name is not found, output a suitable message. Ensure that the search is not case sensitive.

Task 3.4

New members have joined the club. Extend your program code to allow the new members' data to be added to the file.

Extension task

Task 3.5

The club needs to save additional information:

- telephone number
- membership start date

Write new program code to:

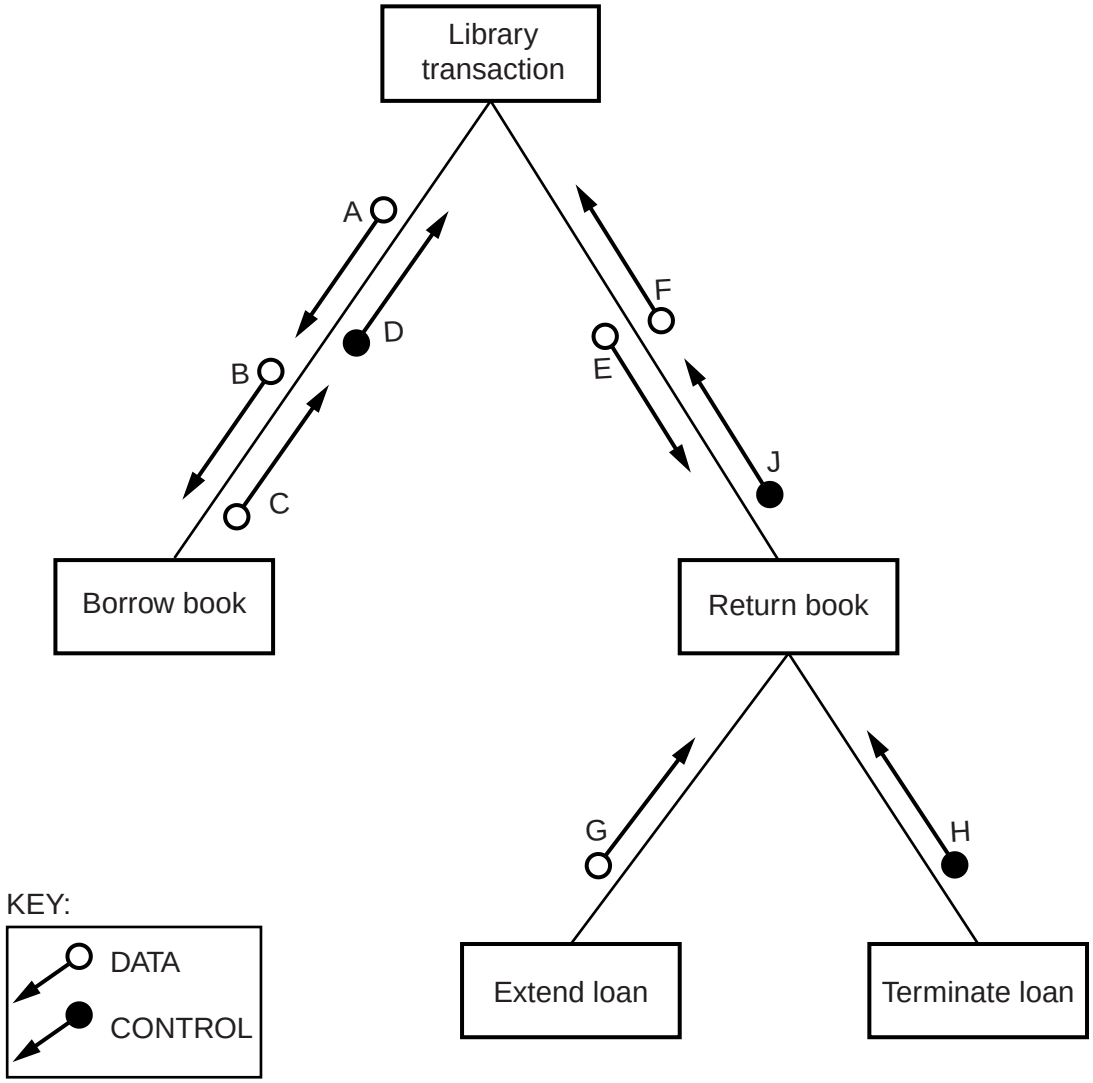
- read the data from the original file for one member
- prompt the user to input the two additional data items
- save all the data for the current member in a new file
- repeat the above steps for each member in the original file

Task 4 – Structure charts

The structure chart illustrates the borrowing and returning of books from a lending library.

All borrowers have a borrower ID. All books have a book ID. When a book is borrowed the due date is recorded and the book marked as unavailable (on-loan).

When a book is due for return, the borrower may either extend or terminate the loan.



Task 4.1

Draw on the chart the symbols that represent:

- the borrower may repeatedly borrow books
- a loan may be either extended or terminated

Task 4.2

Each arrow in the structure chart opposite represents a parameter.

The table below shows the four data items that the nine parameters **A** to **J** pass between modules.

Put a tick (✓) in each of the **nine** parameter columns (**A** to **J**) to show which data item the parameter passes.

Data item	A	B	C	D	E	F	G	H	J
Book ID									
Borrower ID									
Return Date									
Flag Value – available / on-loan									

Appendix

Built-in functions

In each function below, if the function call is not properly formed the function returns an error.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`

Returns the string of length **y** starting at position **x** from **ThisString**
Example: **MID("ABCDEFGH", 2, 3)** will return string **"BCD"**

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`

Returns the leftmost **X** characters from **ThisString**
Example: **LEFT("ABCDEFGH", 3)** will return string **"ABC"**

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`

Returns the rightmost **X** characters from **ThisString**
Example: **RIGHT("ABCDEFGH", 3)** will return string **"FGH"**

`LENGTH(ThisString : STRING) RETURNS INTEGER`

Returns the integer value representing the length of string **ThisString**
Example: **LENGTH("Happy Days")** will return **10**

`CHR(x : INTEGER) RETURNS CHAR`

Returns the character whose ASCII value is **x**
Example: **CHR(87)** will return **'W'**

`ASC(x : CHAR) RETURNS INTEGER`

Returns the ASCII value of character **x**
Example: **ASC('W')** will return **87**

`LCASE(x : CHAR) RETURNS CHAR`

Returns the lower case equivalent character of **x**
Example: **LCASE('W')** will return **'w'**

`UCASE (x : CHAR) RETURNS CHAR`

Returns the upper case equivalent character of **x**
Example: **UCASE('h')** will return **'H'**

`INT(x : REAL) RETURNS INTEGER`

Returns the integer part of **x**
Example: **INT(27.5415)** will return **27**

String operator

`&` operator

Is used to concatenate two strings.
Example: **"Summer" & " " & "Pudding"** produces **"Summer Pudding"**