Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/41**

Paper 4 Further Problem-solving and Programming Skills **October/November 2017**

PRE-RELEASE MATERIAL

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.

DC (NH) 129952/4

Teachers and candidates should read this material prior to the November 2017 examination for 9608 Paper 4.

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.

The practical skills for Paper 4 build on the practical skills covered in Paper 2. We therefore recommend that candidates choose the same high-level programming language for this paper as they did for Paper 2. This will give candidates the opportunity for extensive practice and allow them to acquire sufficient expertise.

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 or vice versa

Candidates will also benefit from using pre-release materials from previous examinations. These are available on the teacher support site.

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

Key focus: Declarative programming

Some clauses in a declarative language are shown.

```
01 male(james).
02 male(alex).
03 male(richard).
04 female(sophie).
05 female(alexandra).
06 female(jane).
07 parent(jane, james).
08 parent(jane, alex).
09 parent(richard, james).
10 parent(richard, alex).
```

Clause	Explanation
01	James is male
04	Sophie is female
07	Jane is a parent of James

TASK 1.1

Add additional clauses to this knowledge base.

TASK 1.2

Write a query to find the mother of a person.

TASK 1.3

Write a query to find the father of a person.

TASK 1.4

Write a query for a sibling of a person. Two people are siblings if they have the same mother and father. A person cannot be their own sibling.

TASK 1.5

Add clauses so that the knowledge base includes parents of parents. We call these ‘grandparents’.

TASK 1.6

Write a query to find the grandparents of a person.

TASK 1.7

Explain the difference between the following two clauses.

- `parent(richard, james).`
- `parent(james, richard).`

TASK 2

Key focus: Low-level programming

The following table shows part of the instruction set for a processor that has one general purpose register, the Accumulator (ACC), and an Index Register (IX).

Instruction		Explanation
Op code	**Operand**	
LDM	#n	Immediate addressing. Load the number n to ACC.
LDD	<address>	Direct addressing. Load the contents of the location at the given address to ACC.
LDI	<address>	Indirect addressing. The address to be used is at the given address. Load the contents of this second address to ACC.
LDX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents of this calculated address to ACC.
LDR	#n	Immediate addressing. Load the number n to IX.
STO	<address>	Store the contents of ACC at the given address.
STX	<address>	Indexed addressing. Form the address from <address> + the contents of the index register. Copy the contents from ACC to this calculated address.
ADD	<address>	Add the contents of the given address to the ACC.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
DEC	<register>	Subtract 1 from the contents of the register (ACC or IX).
JMP	<address>	Jump to the given address.
CMP	<address>	Compare the contents of ACC with the contents of <address>.
CMP	#n	Compare the contents of ACC with number n.
JPE	<address>	Following a compare instruction, jump to <address> if the compare was True.
JPN	<address>	Following a compare instruction, jump to <address> if the compare was False.
LSL	#n	Bits in ACC are shifted n places to the left. Zeros are introduced on the right hand end.
LSR	#n	Bits in ACC are shifted n places to the right. Zeros are introduced on the left hand end.
IN		Key in a character and store its ASCII value in ACC.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

Notes:

# denotes immediate addressing
B denotes a binary number, for example B01001010
& denotes a hexadecimal number, for example &4A

Tasks 2.1 to 2.4 all use one of the following two formats for symbolic addressing.

Format			Example
<label>:	<op code>	<operand>	START: LDA #0
<label>:	<data>		NUM1: B01001010

TASK 2.1

Write a program in assembly language using the op codes from the given instruction set to divide a number by 2 using a logical right shift.

TASK 2.2

Write a program in assembly language using the op codes from the given instruction set to multiply a number by 2 using a logical left shift.

TASK 2.3

Write a program in assembly language using the op codes from the given instruction set to add together a series of numbers stored in an array using indexed addressing.

TASK 2.4

Write a program in assembly language using the op codes from the given instruction set to take three single digit numbers as input, and output the largest.

TASK 3

Key focus: Project management

TASK 3.1

Investigate project management techniques and diagrams that may be used to aid this process.

TASK 3.2

Think of a series of activities involved in completing a complex task and list the activities involved.

TASK 3.3

Complete the following table to show the activities from TASK 3.2, and develop a PERT chart to represent your actions.

Activity	Description	Weeks to complete	Dependent on

BLANK PAGE

BLANK PAGE