Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9608/43**

Paper 4 Further Problem-solving and Programming Skills **October/November 2020**

PRE-RELEASE MATERIAL



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.

This document has **8** pages. Blank pages are indicated.

DC (MB) 207072
© UCLES 2020

Teachers and candidates should read this material prior to the November 2020 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:
    - 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 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 a high-level 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.

Candidates will also benefit from using pre-release materials from previous examinations.
These are available on the School Support Hub.

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.

© UCLES 2020

TASK 1 – Low-level programming

Low-level programming is a type of programming language that uses op codes and operands to create instructions.

The 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 of ACC to this calculated address.
ADD	<address>	Add the contents of the given address to 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.
AND	#n	Bitwise AND operation of the contents of ACC with the operand.
AND	<address>	Bitwise AND operation of the contents of ACC with the contents of <address>.
XOR	#n	Bitwise XOR operation of the contents of ACC with the operand.
XOR	<address>	Bitwise XOR operation of the contents of ACC with the contents of <address>.
OR	#n	Bitwise OR operation of the contents of ACC with the operand.
OR	<address>	Bitwise OR operation of the contents of ACC with the contents of <address>. <address> can be an absolute address or a symbolic address.
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.

© UCLES 2020

TASK 1.1

Write assembly language program code that allows a user to input 5 characters. The characters are not stored.

Label	Op code	Operand	Comment
			// initialise COUNT to 0
LOOP:			// input character
			// increment COUNT
			// is COUNT = MAX ?
			// jump to LOOP if FALSE
			// end program
MAX:	5		
COUNT:			

© UCLES 2020

TASK 1.2

Discuss the purpose of the Index Register and how it can be used to access consecutive memory locations.

TASK 1.3

Write assembly language program code that adds the values stored in four consecutive memory locations starting at `NUMBER` using the Index Register.

Store the final total value in memory location `TOTAL`.

Label	Op code	Operand	Comment
			// initialise Index Register to 0
LOOP:			// load value from NUMBER + contents of Index Register
			// add value to TOTAL
			// increment Index Register
			// increment COUNT
			// is COUNT = MAX ?
			// jump to LOOP if FALSE
			// end program
MAX:	4		
COUNT:	0		
NUMBER:	23		
	17		
	38		
	13		
TOTAL:	0		

TASK 1.4

The assembly language instruction set given has the op code `STX`. Discuss the purpose of this op code.

TASK 1.5

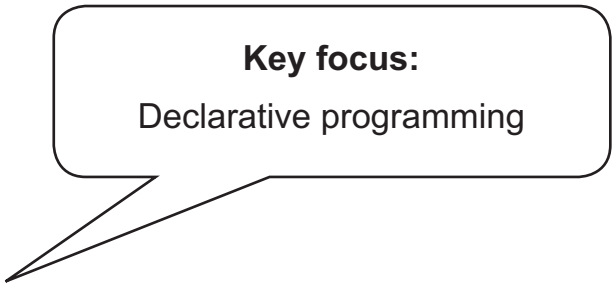
Amend your solution to **TASK 1.1** to allow the program to store each of the characters input into separate, consecutive memory locations starting at the memory locations labelled `CHARACTER`.

TASK 2 – Declarative programming

A knowledge base contains information about students in a class, the colours they like and the colours they do not like. A declarative programming language is used to query the knowledge base.

Some clauses in the knowledge base are shown.

```
01 person(luke).
02 person(alice).
03 person(taylor).
04 person(nadia).
05 colour(blue).
06 colour(red).
07 colour(green).
08 colour(yellow).
09 likes_colour(alice, yellow).
10 likes_colour(alice, blue).
11 dislikes_colour(taylor, red).
12 dislikes_colour(nadia, green).
```



Clause	Explanation
01	Luke is a person
07	Green is a colour
09	Alice likes the colour yellow
11	Taylor does not like the colour red

TASK 2.1

Two new students are joining the class: Mehrdad and Nigel. They need to be added to the knowledge base.

Four further colours: pink, orange, purple and black need to be added to the knowledge base.

Write clauses to add the two new students and the new colours to the knowledge base.

TASK 2.2

Add a clause that states Nadia likes the colour red.

TASK 2.3

Add a clause that states Mehrdad does not like the colour pink.

TASK 2.4

Write a goal to find all the colours that a person likes.

© UCLES 2020

TASK 3 – Abstract Data Type (ADT)

Key focus:
Linked lists

A linked list is an Abstract Data Type.

A linked list is used to store data in a linear structure.

TASK 3.1

Discuss what a node and a pointer are in terms of a linked list.

TASK 3.2

A company has a list of destinations that are visited as part of a round the world holiday.

The destinations are:

- Paris, France
- Rome, Italy
- New Delhi, India
- Kuala Lumpur, Malaysia
- Wellington, New Zealand
- New York, USA

The destinations are stored in a linked list in the order shown.

Draw a diagram to represent the data as a linked list.

Use the symbol Ø to represent the null pointer.

TASK 3.3

A further destination is added after New York; this destination is Reykjavik, Iceland.

Add the new destination to the diagram of your linked list.

TASK 3.4

Discuss how a node would be removed from the linked list.

TASK 3.5

Write **program code** to declare the linked list, using an array.

TASK 3.6

Extend your **program code** by writing a subroutine that adds a new destination to the end of your linked list.

TASK 3.7

Extend your **program code** by writing a subroutine to delete the destination node entered by the user from the linked list.

TASK 3.8

Discuss other linked list operations that could be implemented.

Write **program code** to implement the operation(s) you discuss.

Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cambridgeinternational.org after the live examination series.

Cambridge Assessment International Education is part of the Cambridge Assessment Group. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which itself is a department of the University of Cambridge.

© UCLES 2020