Cambridge International A Level

Cambridge International Examinations
Cambridge International Advanced Level

CANDIDATE NAME	
CENTRE NUMBER	
CANDIDATE NUMBER	

4530773208

COMPUTER SCIENCE **9608/42**

Paper 4 Further Problem-solving and Programming Skills **October/November 2016**

2 hours

Candidates answer on the Question Paper.

No Additional Materials are required.

No calculators allowed.

READ THESE INSTRUCTIONS FIRST

Write your Centre number, candidate number and name in the spaces at the top of this page.
Write in dark blue or black pen.
You may use an HB pencil for any diagrams, graphs or rough working.
Do not use staples, paper clips, glue or correction fluid.
DO **NOT** WRITE IN ANY BARCODES.

Answer **all** questions.
No marks will be awarded for using brand names of software packages or hardware.

At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **17** printed pages and **3** blank pages.

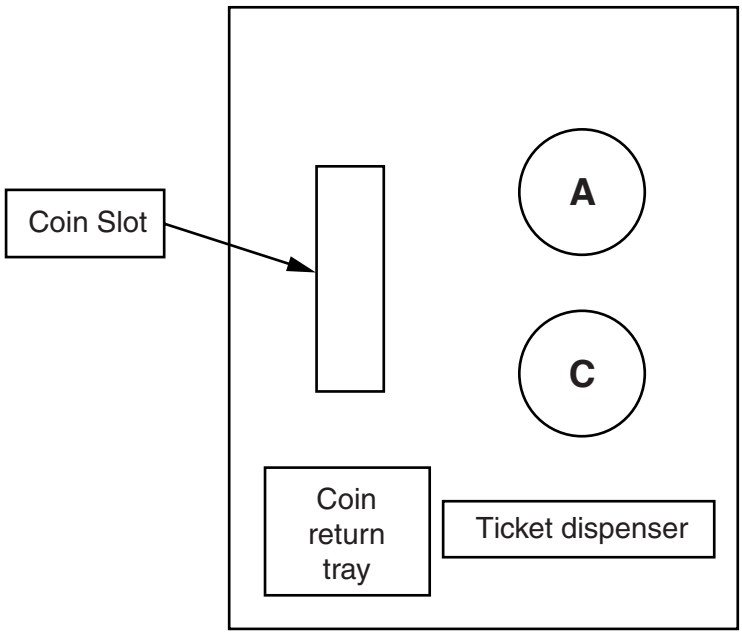
© UCLES 2016

CAMBRIDGE International Examinations

1 The ticket machine in the following diagram accepts the following coins: 10, 20, 50 and 100 cents.

The ticket machine has:

- a slot to insert coins
- a tray to return coins
- a ticket dispenser
- two buttons:
    - button **A** (Accept)
    - button **C** (Cancel)



When the user has inserted as many coins as required, they press button **A** to print the ticket.

To cancel the transaction, the user can press button **C**. This makes the machine return the coins.

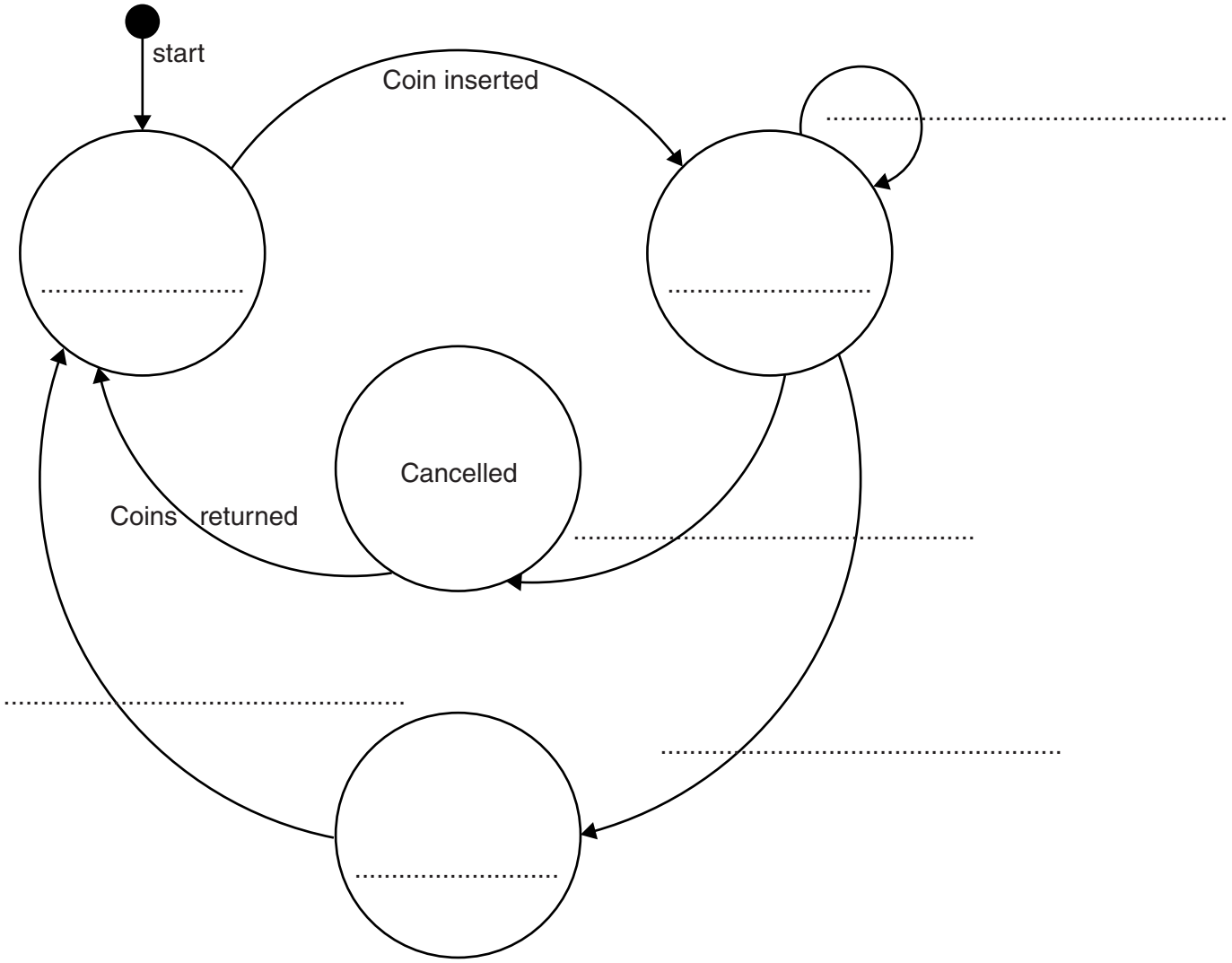
Invalid coins have no effect.

© UCLES 2016

The following state transition table shows the transition from one state to another of the ticket machine:

Current state	Event	Next state
Idle	Coin inserted	Counting
Counting	Coin inserted	Counting
Counting	Button **C** pressed	Cancelled
Cancelled	Coins returned	Idle
Counting	Button **A** pressed	Accepted
Accepted	Ticket printed	Idle

(a) Complete the state-transition diagram.



[7]

(b) A company wants to simulate the use of a ticket machine. It will do this with object-oriented programming (OOP).

The following diagram shows the design for the class `TicketMachine`. This includes its attributes and methods.

TicketMachine

`Amount : INTEGER // total value of coins inserted in cents`
`State : STRING // "Idle", "Counting", "Cancelled"`
`// or "Accepted"`

`Create() // method to create and initialise an object`
`// if using Python use __init__`
`SetState() // set state to parameter value`
`// and output new state`
`StateChange() // insert coin or press button,`
`// then take appropriate action`
`CoinInserted() // parameter is a string`
`// change parameter to integer`
`// and add coin value to Amount`
`ReturnCoins() // output Amount, then set Amount to zero`
`PrintTicket() // print ticket, then set Amount to zero`

Write **program code** for the following methods.

Programming language .....

(i) `Create()`

.....

.....

.....

.....

.....[3]

(ii) `SetState()`

.....

.....

.....

.....[2]

(iii) `ReturnCoins()`

.....

.....

.....

..... [2]

(iv) Each coin inserted must be one of the following: 10, 20, 50 or 100 cents.

Write **program code** for a function `ValidCoin(s : STRING)` that returns:

- `TRUE` if the input string is one of `"10"`, `"20"`, `"50"` or `"100"`
- `FALSE` otherwise

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....[3]

(v) Write **program code** for the method `CoinInserted()`

.....

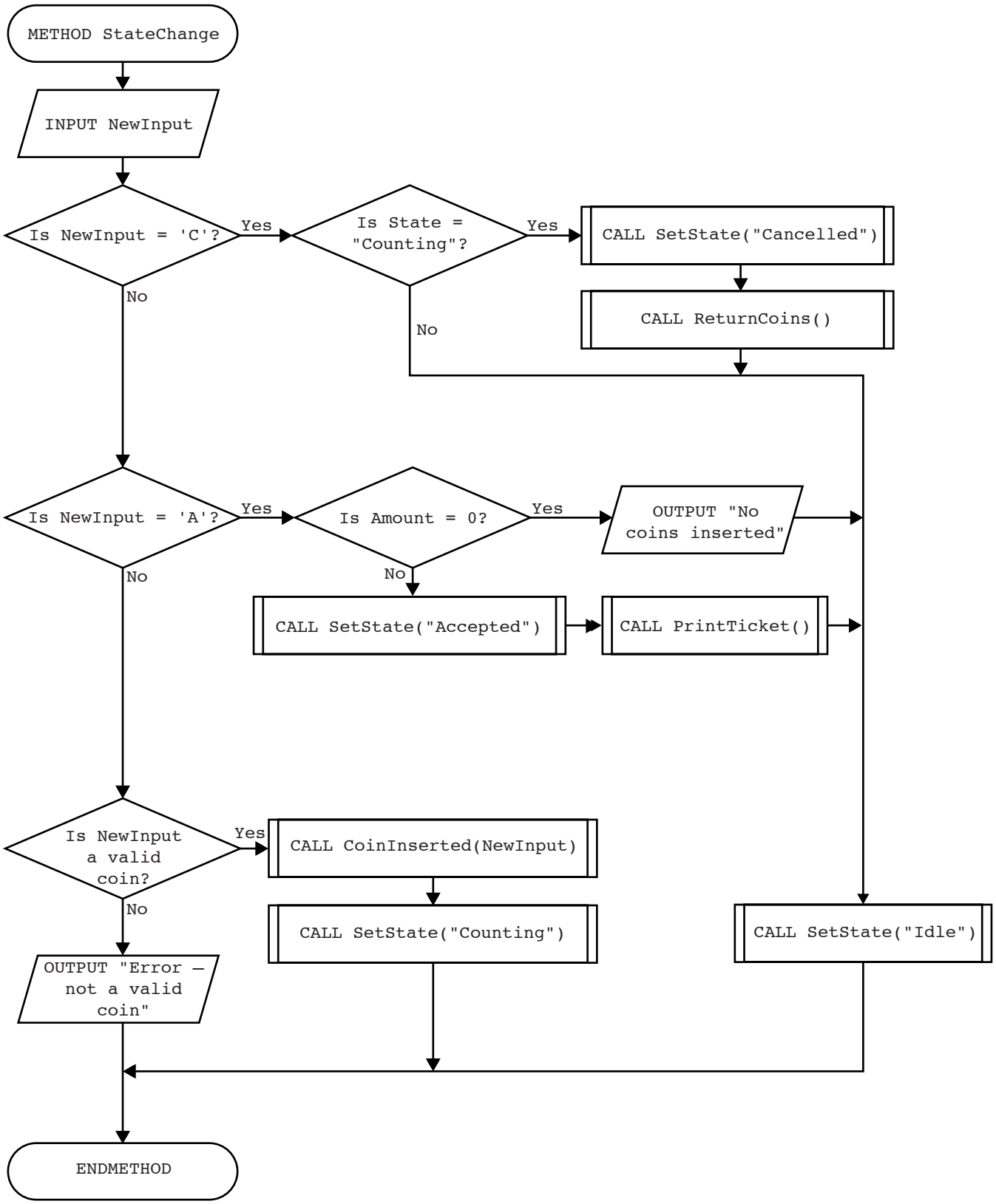
.....

.....

.....

.....[2]

(vi) Convert the flowchart to **program code** for the method `StateChange()`.
Use the attributes and methods in the original class definition and the `ValidCoin()` function from **part (iv)**.



Programming language ....

....[12]

(vii) The company needs to write a program to simulate a parking meter. The program will create an object with identifier `ParkingMeter`, which is an instance of the class `TicketMachine`.

The main program design is:

```
instantiate ParkingMeter (create and initialise ParkingMeter)
loop forever (continually use ParkingMeter)
   call StateChange() method
end loop
```

Write **program code** for the main program.

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....[4]

(c) It is possible to declare attributes and methods as either public or private.

A programmer has modified the class design for `TicketMachine` as follows.

TicketMachine
PRIVATE Amount : INTEGER State : STRING
PUBLIC Create() StateChange() PRIVATE SetState() CoinInserted() ReturnCoins() PrintTicket()

(i) Describe the effects of declaring the `TicketMachine` attributes as private.

.....

.....

.....

.....[2]

(ii) Describe the effects of declaring two methods of the class as public and the other four as private.

.....

.....

.....

.....[2]

2 Commercial software usually undergoes alpha testing and beta testing.

Distinguish between the two types of testing by stating:

- who does the testing
- when the testing occurs
- the specific purpose of each type of testing

(i) Alpha testing

Who .....

.....

When .....

.....

Purpose .....

.....[3]

(ii) Beta testing

Who .....

.....

When .....

.....

Purpose .....

.....[3]

3 **(a)** The numerical difference between the ASCII code of an upper case letter and the ASCII code of its lower case equivalent is 32 denary (32_{10}).

For example, `'F'` has ASCII code 70 and `'f'` has ASCII code 102.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0

The bit patterns differ only at bit number 5. This bit is 1 if the letter is lower case and 0 if the letter is upper case.

(i) A program needs a mask to ensure that a letter is in **upper case.**

Write the binary pattern of the mask in the space provided in the table below.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0
Mask								

Give the bit-wise operation that needs to be performed using the mask and the ASCII code.

.....[2]

(ii) A program needs a mask to ensure that a letter is in **lower case**.

Write the binary pattern of the mask in the space provided in the table below.

	Bit number							
	7	6	5	4	3	2	1	0
ASCII code	ASCII code in binary							
70	0	1	0	0	0	1	1	0
102	0	1	1	0	0	1	1	0
Mask								

Give the bit-wise operation that needs to be performed using the mask and the ASCII code.

.....[2]

The following table shows part of the instruction set for a processor which 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 given 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 into IX.
STO	<address>	Store the contents of ACC at the given address.
INC	<register>	Add 1 to the contents of the register (ACC or IX).
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>.
OUT		Output to the screen the character whose ASCII value is stored in ACC.
END		Return control to the operating system.

A programmer is writing a program that will output the first character of a string in upper case and the remaining characters of the string in lower case.

The program will use locations from address `WORD` onwards to store the characters in the string. The location with address `LENGTH` stores the number of characters that make up the string.

The programmer has started to write the program in the following table. The comment column contains descriptions for the missing program instructions.

(b) Complete the program using op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// get first character of WORD
			// ensure it is in upper case using MASK1
			// output character to screen
			// increment index register
			// load 1 into ACC
			// store in COUNT
LOOP:			// load next character from indexed address WORD
			// make lower case using MASK2
			// output character to screen
			// increment COUNT starts here
			// is COUNT = LENGTH ?
			// if FALSE, jump to LOOP
			// end of program
COUNT:			
MASK1:			// bit pattern for upper case
MASK2:			// bit pattern for lower case
LENGTH:		4	
WORD:		B01100110	// ASCII code in binary for 'f'
		B01110010	// ASCII code in binary for 'r'
		B01000101	// ASCII code in binary for 'E'
		B01000100	// ASCII code in binary for 'D'

[12]

© UCLES 2016

[Turn over

Question 4 begins on page 15.

4 Circle the programming language that you have studied:

Visual Basic (console mode) Python Pascal Delphi (console mode)

(a) (i) Name the programming environment you have used when typing in program code.

.....

.....

List **three** features of the editor that helped you to write program code.

1 .....

.....

2 .....

.....

3 .....

.....[3]

(ii) Explain when and how your programming environment reports a syntax error.

When .....

.....

.....

How .....

.....

.....[2]

(iii) The table shows a module definition for `BubbleSort` in three programming languages.

Study **one** of the examples. Indicate your choice by circling A, B or C:

A B C

A) Python

```
01  def BubbleSort(SList, Max):
02     NoMoreSwaps = False
03     while NoMoreSwaps == False:
04        NoMoreSwaps = True
05        for i in (Max - 1):
06           if SList[i] > SList[i + 1]:
07              NoMoreSwaps = True
08              Temp = SList[i]
09              SList[i] = SList[i + 1]
10              SList[i + 1] = Temp
```

B) Pascal/Delphi

```
01  PROCEDURE BubbleSort(VAR SList : ARRAY OF INTEGER; Max : INTEGER);
02  VAR NoMoreSwaps : BOOLEAN; i, Temp : INTEGER;
03  BEGIN
04     REPEAT
05        NoMoreSwaps := TRUE;
06        FOR i := 1 TO (Max - 1)
07           IF SList[i] > SList[i + 1]
08              THEN
09                 BEGIN
10                    NoMoreSwaps := TRUE;
11                    Temp := SList[i];
12                    SList[i] := SList[i + 1];
13                    SList[i + 1] := Temp;
14                 END;
15     UNTIL NoMoreSwaps;
16  END;
```

C) Visual Basic

```
01  Sub BubbleSort(ByRef SList() As Integer, ByVal Max As Integer)
02     Dim NoMoreSwaps As Boolean, i, Temp As Integer
03        Do
04           NoMoreSwaps = True
05           For i : 0 To (Max - 1)
06              If SList(i) > SList(i + 1) Then
07                 NoMoreSwaps = True
08                 Temp = SList(i)
09                 SList(i) = SList(i + 1)
10                 SList(i + 1) = Temp
11              End If
12           Next
13        Loop Until (NoMoreSwaps = True)
14  End Sub
```

© UCLES 2016

The programming environment reported a syntax error in the `BubbleSort` code.

State the line number .....

Write the correct code for this line.

.....[2]

(b) (i) State whether programs written in your programming language are compiled or interpreted.

.....

.....[1]

(ii) A programmer corrects the syntax error and tests the function. It does not perform as expected. The items are not fully in order.

State the type of error .....

Write the line number where the error occurs.

.....

Write the correct code for this line.

.....[2]

(iii) State the programming environment you have used when debugging program code.

.....

Name **two** debugging features and describe how they are used.

1 .....

.....

.....

.....

2 .....

.....

.....

.....[4]

BLANK PAGE

© UCLES 2016

BLANK PAGE

© UCLES 2016

BLANK PAGE

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 International Examinations Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at www.cie.org.uk after the live examination series.

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