Cambridge International **A Level**

Cambridge International Examinations
Cambridge International Advanced Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

8118378603

COMPUTER SCIENCE **9608/41**

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.

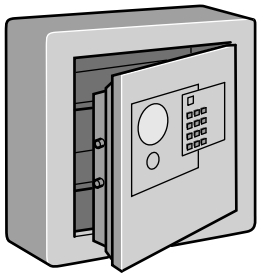
At the end of the examination, fasten all your work securely together.
The number of marks is given in brackets [ ] at the end of each question or part question.

The maximum number of marks is 75.

This document consists of **16** printed pages.

CAMBRIDGE International Examinations

1 A user can lock a safety deposit box by inputting a 4-digit code. The user can unlock the box with the same 4-digit code.



There is a keypad on the door of the safety deposit box. The following diagram shows the keys on the keypad.

1	2	3
4	5	6
7	8	9
R	0	Enter

Initially, the safety deposit box door is open and the user has not set a code.

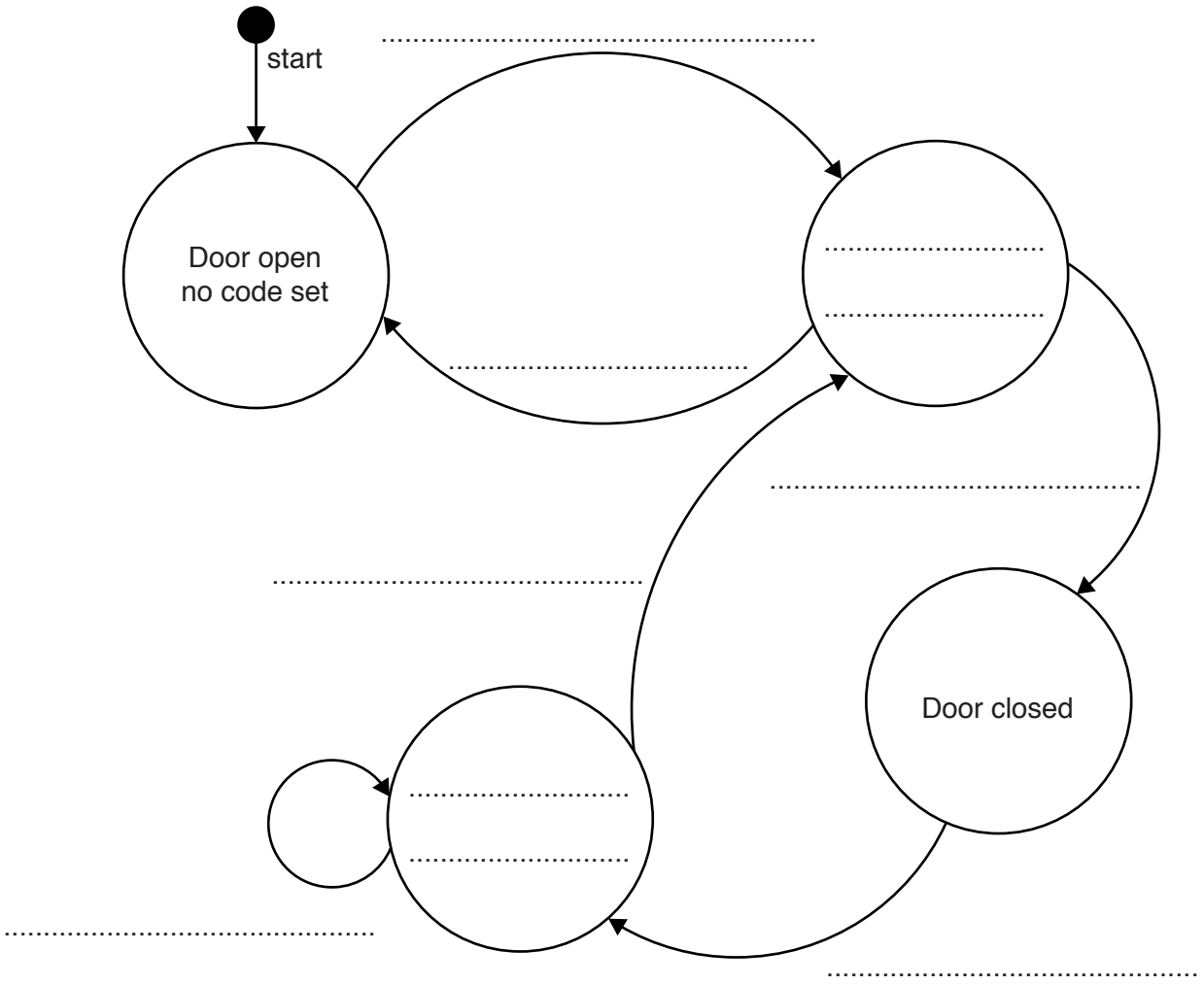
The operation of the safety deposit box is as follows:

A) To set a new code the door must be open. The user chooses a 4-digit code and sets it by pressing the numerical keys on the keypad, followed by the Enter key. Until the user clears this code, it remains the same. (See point E below)
B) The user can only close the door if the user has set a code.
C) To lock the door, the user closes the door, enters the set code and presses the Enter key.
D) To unlock the door, the user enters the set code. The door then opens automatically.
E) The user clears the code by opening the door and pressing the R key, followed by the Enter key. The user can then set a new code. (See point A above)

The following state transition table shows the transition from one state to another of the safety deposit box:

Current state	Event	Next state
Door open, no code set	4-digit code entered	Door open, code set
Door open, code set	R entered	Door open, no code set
Door open, code set	Close door	Door closed
Door closed	Set code entered	Door locked
Door locked	Set code entered	Door open, code set
Door locked	R entered	Door locked

(a) Complete the state-transition diagram.



[7]

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

The following diagram shows the design for the class `SafetyDepositBox`. This includes the properties and methods.

SafetyDepositBox

`Code : STRING // 4 digits`
`State : STRING // "Open-NoCode", "Open-CodeSet", "Closed"`
`// or "Locked"`

`Create() // method to create and initialise an object`
`// if using Python use __init__`
`Reset() // clears Code`
`SetState() // set state to parameter value`
`// and output new state`
`SetNewCode() // sets Code to parameter value`
`// output message and new code`
`StateChange() // reads keypad and takes appropriate action`

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

Programming language .....

(i) `Create()`

.....

.....

.....

.....

..... [3]

(ii) `Reset()`

.....

.....

..... [2]

(iii) `SetState()`

.....

.....

.....

..... [2]

(iv) `SetNewCode()`

.....

.....

.....

..... [2]

(v) The user must enter a 4-digit code.

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

- `TRUE` if the input string `s` consists of exactly 4 digits
- `FALSE` otherwise

Programming language .....

.....

.....

.....

.....

.....

.....

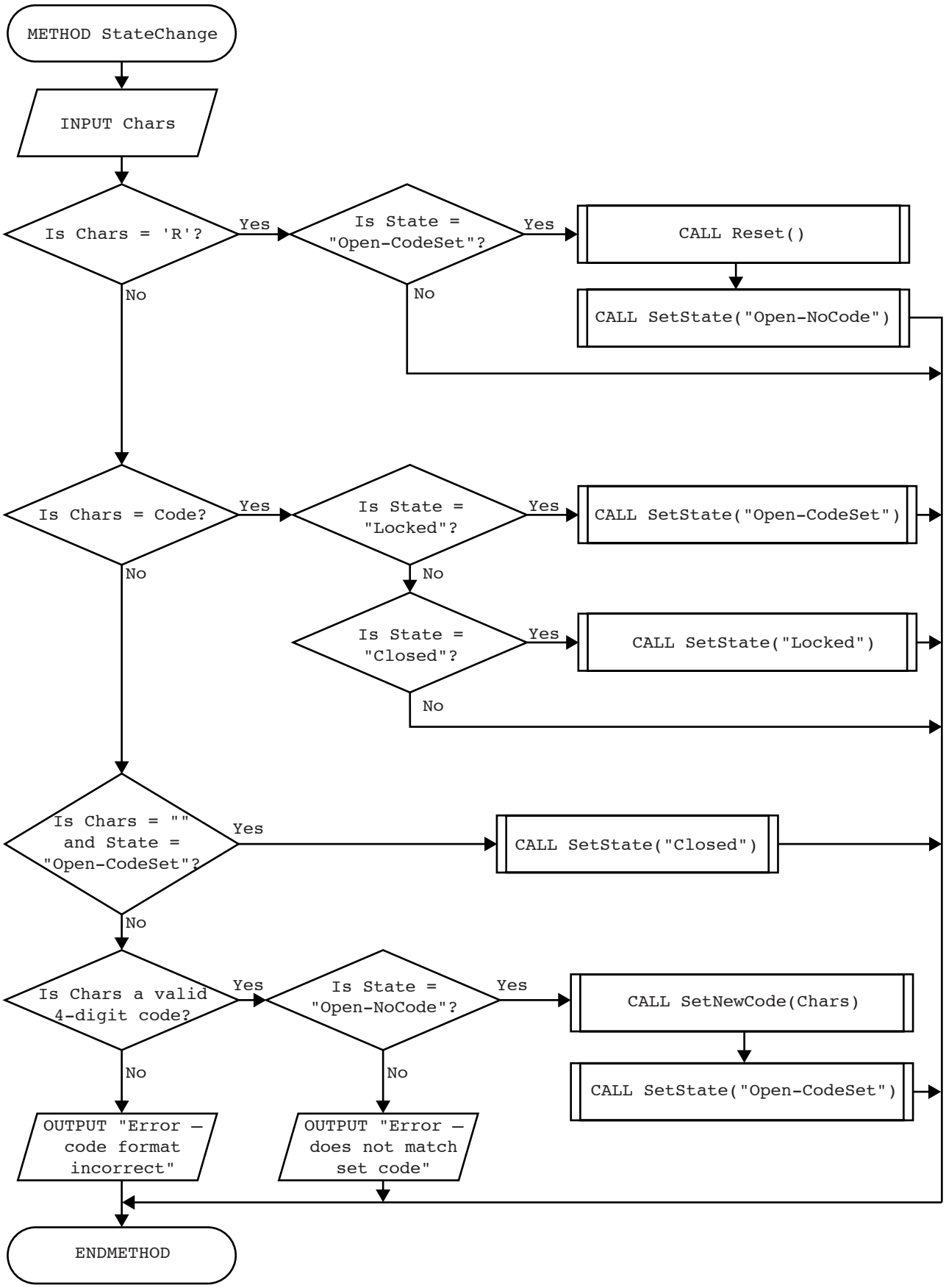
.....

.....

.....

..... [4]

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



Programming language ....

....[12]

(vii) The company needs to write a program to simulate a safety deposit box. The program will create an object with identifier `ThisSafe`, which is an instance of the class `SafetyDepositBox`.

The main program design is:

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

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

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [4]

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

The programmer has modified the class design for `SafetyDepositBox` as follows:

SafetyDepositBox
PRIVATE Code : STRING State : STRING
PUBLIC Create() StateChange() PRIVATE Reset() SetState() SetNewCode()

(i) Describe the effects of declaring the `SafetyDepositBox` properties as private.

.....

.....

.....

..... [2]

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

.....

.....

.....

..... [2]

2 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]

Question 2 continues on page 12.

(iii) The table shows a module definition for `BinarySearch` 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 BinarySearch(List, Low, High, SearchItem):
02    Index = -1
03    while (Index == -1) AND (Low <= High):
04       Middle = (High + Low) // 2
05       if List[Middle] == SearchItem:
06          Index = Middle
07       elif List[Middle] < SearchItem:
08          Low = Middle + 1
09       else:
10          High = Middle - 1
11    return(Middle)
```

B) Pascal/Delphi

```
01 FUNCTION BinarySearch(VAR List : ARRAY OF INTEGER; Low, High,
                                        SearchItem : INTEGER) : INTEGER;
02 VAR Index, Middle : INTEGER;
03 BEGIN
04    Index := -1;
05    WHILE (Index = -1) & (Low <= High) DO
06       BEGIN
07          Middle := (High + Low) DIV 2;
08          IF List[Middle] = SearchItem
09             THEN Index := Middle
10             ELSE IF List[Middle] < SearchItem
11                     THEN Low := Middle + 1
12                     ELSE High := Middle - 1;
13       END;
14    Result := Middle;
15 END;
```

C) Visual Basic

```
01 Function BinarySearch(ByRef List() As Integer, ByVal Low As Integer,
         ByVal High As Integer, ByVal SearchItem As Integer) As Integer
02    Dim Index, Middle As Integer
03    Index = -1
04    Do While (Index = -1) & (Low <= High)
05       Middle = (High + Low) \ 2
06       If List(Middle) = SearchItem Then
07          Index = Middle
08       ElseIf List(Middle) < SearchItem Then
09          Low = Middle + 1
10       Else
11          High = Middle - 1
12       End If
13    Loop
14    BinarySearch = Middle
15 End Function
```

The programming environment reported a syntax error in the `BinarySearch` 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 when the search item is not in the list.

State the type of error: .....

Write down 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]

3 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).
DEC	<register>	Subtract 1 from 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.
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 outputs a string, first in its original order and then in reverse order.

The program will use locations starting at address NAME to store the characters in the string. The location with address MAX stores the number of characters that make up the string.

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

Complete the program using op codes from the given instruction set.

Label	Op code	Operand	Comment
START:			// initialise index register to zero
			// initialise COUNT to zero
LOOP1:			// load character from indexed address NAME
			// output character to screen
			// increment index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP1
REVERSE:			// decrement index register
			// set ACC to zero
			// store in COUNT
LOOP2:			// load character from indexed address NAME
			// output character to screen
			// decrement index register
			// increment COUNT starts here
			// is COUNT = MAX ?
			// if FALSE, jump to LOOP2
			// end of program
COUNT:			
MAX:	4		
NAME:	B01000110		// ASCII code in binary for 'F'
	B01010010		// ASCII code in binary for 'R'
	B01000101		// ASCII code in binary for 'E'
	B01000100		// ASCII code in binary for 'D'

[15]

[Turn over

4 Commercial software usually undergoes acceptance testing and integration 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) Acceptance testing

Who .....

.....

When .....

.....

Purpose .....

.....[3]

(ii) Integration testing

Who .....

.....

When .....

.....

Purpose .....

.....[3]