CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the May/June 2015 series

9691 COMPUTING

9691/21 Paper 2 (Written Paper), maximum raw mark 75

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

1 (a) (i) 66

(ii) error

(iii) 'C' (accept without quotes) **[1]**

(b) `Letter15 ← CHAR(ASCII('A') + 14)` **[2]**

Completely correct – 2 marks
Single error of (not 14) scores 1 mark

(c) (i)
- letter A-Z have increasing ASCII codes
- the ASCII values of the two characters are compared
- the character with the smaller value is the first character / the character with the larger value is the second character **[2]**

(ii)
- ASCII codes of the characters are compared in turn …
- from left hand side / start of each word
- … until two characters are different
- the lower code value determines the first word
- if 2 words are the same when one ends …
- … this is the first word **[4]**

(iii) Mark as follows:
- Function header (ignore data type) & termination
- Data types for parameter and return value
- Change letter to ASCII
- Add 32
- Change ASCII code to letter
- Return value

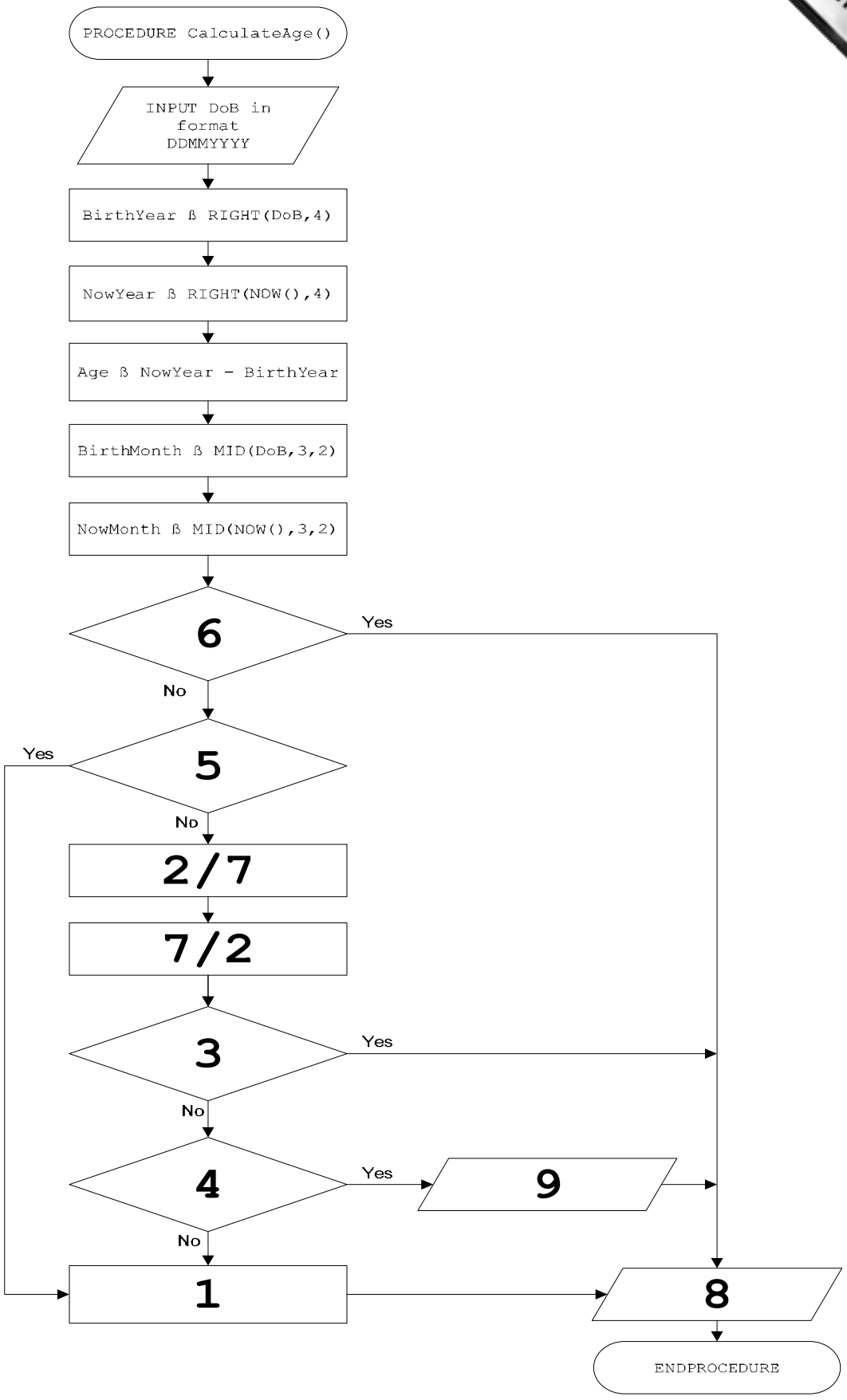
Example pseudocode

```
FUNCTION LowerCase(Letter : CHARACTER) RETURNS CHARACTER
   DECLARE LetterCode : INTEGER
   LetterCode ← ASCII(Letter) + 32
   Letter ← CHAR(LetterCode)
   RETURN Letter
ENDFUNCTION
```

[6]

2 (i) `"01072015"` **[1]**

(ii)



1 mark for each box except 2/7 are 1 mark for both. **[8]**

(iii) Five dates to cover the following cases:
- Birth month before current month
- Birth month after current month
- Birth month equal to current month + birth day before current day
- Birth month equal to current month + birth day after current day
- Birth month equal to current month + birth day equal to current day **[5]**

3 (a) (i) *Mark as follows:*
- *correct index range*
- *correct data type*

Example Pascal:

```
VAR Letters : ARRAY[0..25] OF INTEGER;
```
[2]

(ii) 0
Do not accept "0" **[1]**

(iii) *Mark as follows:*
- *correct loop from 0 to 25 (accept REPEAT or WHILE loops that work)*
- *assignment of initial value to array element (allow ft from part **(ii)**)*

Example Pascal

```
FOR i := 0 TO 25 DO
   Letters[i] := 0;
```
[2]

(b) (i)
```
WHILE NOT EOF(MessageText)
 ::
// calculate index using ASCII function from Question 1
   Index ← ASCII(NextLetter) - ASCII('A')
  // increment relevant frequency total in Letters array
   Letters[Index] ← Letters[Index] + 1
```
[3]

(ii)
- returns a Boolean value
- checks whether it reached a marker written to the file …
- immediately after the last character **[max 2]**

(No marks for "End Of File")

(c) (i) *Mark as follows:*

- *parameter*
- *returns data type*
- *declaration of local variable(s)*
- *Initialisation(s)*
- *loop*
- *Boolean statement*
- *updating of largest so far*
- *store index of where largest so far was found*
- *return index of most frequent letter*

Example answer:

```
FUNCTION MostFrequentLetterIndex(Letters : ARRAY OF INTEGER)
                                           RETURNS INTEGER
   DECLARE Index : INTEGER
   DECLARE LargestSoFar : INTEGER
   DECLARE i : INTEGER
   LargestSoFar ← 0
   Index ← -1    // reject a value within 0 to 25
   FOR i ← 0 TO 25
      IF Letters[i] > LargestSoFar
         THEN
           LargestSoFar ← Letters[i]
           Index ← i
      ENDIF
   ENDFOR
   RETURN Index
ENDFUNCTION
```

[max 8]

(ii) `MostFrequentLetter ← CHAR(MostFrequentLetterIndex() + 65)` **[1]**

(iii) `Displacement ← ASCII(MostFrequentLetter) – ASCII('E')` **[1]**

(d) (i)

x	**y**	**z**	**w**	**OUTPUT**
"E"	69	72	"H"	"H"
"B"	66	69	"E"	"E"
"I"	73	76	"L"	"L"
"M"	77	80	"P"	"P"

1 mark per column (first three) – 1 mark last two columns **[4]**

(ii) Converts an encrypted message into plain text **[1]**

(iii) Any **one** from:
- Annotation / comments
- Keywords in capitals

(iv) Meaningful variable names
Indentation **[2]**

(e) (i) Any example of a syntax error such as:
mis-spelling of keyword
mismatched brackets **[1]**

(ii) syntax error
When: during compilation // during code entry into Integrated Development Environment
How: translator diagnostics / compiler error messages // IDE highlights error **[2]**

(iii) (The logic of) the method of solution was not correct
Or by example **[1]**

(iv) logic error
When: during testing / execution
How: when expected results don't match actual results **[2]**

(f) (i)
```
03 FOR i ← 0 TO 25
04   Used[i] ← FALSE
```
[2]

(ii)
```
06 FUNCTION RandomCode () RETURNS INTEGER
07    REPEAT
08       Code ← Random(25)
09    UNTIL Used[Code] = FALSE
10    Used[Code] ← TRUE
11    RETURN Code
12 ENDFUNCTION
```
[4]

(iii)
```
13 // main program
14 // calculate and store unique random letters
15 // in second column of array LetterGrid
16 FOR i ← 0 TO 25
17    LetterGrid[i,2] ← CHAR(65 + RandomCode())
18 ENDFOR
```
[2]

(iv)
- check contents of `LetterGrid` array
- every letter is there exactly once in second column **[2]**