CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the October/November 2015 series

9691 COMPUTING

9691/22 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 October/November 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

CAMBRIDGE
International Examinations

1 (a)

Field	Identifier	Data type	Example of input data	Field size (in bytes)	Marks
Title	Title	STRING (not text)	How to solve it	30 approx. (accept a range)	1
Author	Author	STRING (not text)	G Polya	20 approx. (accept a range)	
International Standard Book Number	ISBN	STRING / LONGINT	9780691119663	13 minimum	1
Number of pages	NumberOfPages	INTEGER	253	4	1
Price($)	BookPrice	CURRENCY/FLOAT /SINGLE/REAL /DOUBLE/DECIMAL	12.50	8/16/32/64	1
Date started to read book	DateStarted	DATE / REAL (Accept STRING)	28032012	8	1
Date finished reading book	DateFinished	DATE / REAL (Accept STRING)	17052012	8	
Paperback?	IsPaperback	BOOLEAN	TRUE	1/2	1
Rating (Range 0 to 5)	Rating	INTEGER/BYTE /CHARACTER /STRING(1)	4	4 / 1	1

[max 5]

(b) Mark as follows:
1 mark for correct record header
1 mark for correct definition terminator
1 mark for the first 5 fields defined correctly for language
1 mark for the remaining 4 fields defined correctly for language

Do not accept pseudocode
Field names must be as given, but ignore capitalisation/spaces
Declared program language must match code given
Ignore field sizes and data type
If misused DIM in VB, penalise once
If statement separators missed, penalise once

Example Pascal:

```
TYPE   BookRecordType = RECORD
       Title: STRING[20];
       Author: STRING[20];
       ISBN: STRING[13];
       NumberOfPages: INTEGER;
       BookPrice: Currency;
       DateStarted: TDateTime;
       DateFinished: TDateTime;
       IsPaperback: Boolean;
       Rating: INTEGER;
END;
```
[4]

(c) – set up a dummy record // assign each field a dummy value // use a constructor
– …and store this in every element of the array // loop 100 times

Accept code [2]

(d) 1 mark per point below (marks are for method)
- Record size ~ 80bytes
- * 10 (number of records)
- + 10%

Divide by 1024 (do not accept division by 1000) [4]

(e) Mark as follows:
- Open file BookData.DAT
- … for writing/output/append
- test for book data // test this is not a dummy record
- write record to file
- correctly working loop (FOR/WHILE/REPEAT)
- Close file BookData.DAT (or channel number)

Example pseudocode:

```
OPENFILE BookData.DAT FOR WRITING
i ← 1
WHILE i <= 100
   IF Book[i].Title > "" // accept any field and its dummy value
      THEN
         WRITE record to FILE
   ENDIF
   i ← i + 1
ENDWHILE
CLOSEFILE BookData.DAT
```

[Max 5]

(f)
- EOF returns TRUE or FALSE
- Depending on whether it found the marker at the end of the file [2]

(g) (i) Mark as follows:
- initial value of `TopRatingSoFar` (outside loop)
- loop
- compare rating
- update rating if appropriate
- keep note of title/array index
- output top title only
- correct field notation

Example pseudocode:

```
TopRatingSoFar ← 0
i ← 0
REPEAT
   i ← i + 1
   IF TopRatingSoFar < Book[i].Rating
      THEN
         TopRatingSoFar ← Book[i].Rating
         TopBookTitle ← Book[i].Title
   ENDIF
UNTIL Book[i].Title = ""
OUTPUT TopBookTitle
```

Alternative answer:

```
TopRatingSoFar ← 0
FOR i ← TO 100
   IF TopRatingSoFar < Book[i].Rating
      THEN
         TopRatingSoFar ← Book[i].Rating
         TopBookTitle ← Book[i].Title
   ENDIF
ENDFOR
OUTPUT TopBookTitle
```

[max 6]

(ii)
- first loop to find highest rating
- second loop to output relevant titles

[2]

2 (a) (i) Mark as follows:
- parameter
- Return data type
- Correctly formed CASE statement (including the end)
- with all cases present (characters in quotes)
- ELSE clause
- Return of value (implied)

Example PASCAL:

```
FUNCTION NumeralValue(Letter : CHAR) : INTEGER;
   BEGIN
      CASE Letter OF
         'M': NumeralValue := 1000;
         'D': NumeralValue := 500;
         'C': NumeralValue := 100;
         'L': NumeralValue := 50;
         'X': NumeralValue := 10;
         'V': NumeralValue := 5;
         'I': NumeralValue := 1;
      ELSE
         NumeralValue := -1;
      END;
   END;
```

[max 5]

(ii)

`Letter`	**Expected result**	**Type of data** (normal, borderline or invalid)
`'D'`	500	normal
`'V'`	5	normal
`'I'`	1	normal
`'Y'`	–1 (Do not accept Error)	invalid

1 mark for 3 rows of normal data (Do not accept borderline)
1 mark for –1
1 mark for invalid [3]

(b) Mark as follows:

1 mark per column (2 to 6)
If zero marks then mark by row

RomanNumber	Denary	i	ThisLetter	ThisNumber	OUTPUT
"MDCLI"	0				
	1000	1	'M'	1000	
	1500	2	'D'	500	
	1600	3	'C'	100	
	1650	4	'L'	50	
	1651	5	'I'	1	1651

Ignore quotes [5]

(c) (i) Meaningful variable names
Capitalisation of keywords
Use of (library/built-in) functions
Accept empty lines
Do not accept white space
Camel case on its own is too vague [max 2]

(ii) 1 mark per line of pseudocode correctly written in the high-level language chosen:

```
RomanNumber ← "MDCLI"

Denary ← 0

FOR i ← 1 TO LENGTH(RomanNumber)
ThisLetter ← MID(RomanNumber, i, 1)
ThisNumber ← NumeralValue(ThisLetter)
Denary ← Denary + ThisNumber
ENDFOR

OUTPUT Denary (ignore any message)
```

Example Pascal:

```
RomanNumber := 'MDCLI';
Denary := 0;

FOR i := 1 TO LENGTH(RomanNumber) DO
   BEGIN
      ThisLetter := MIDSTR(RomanNumber, i, 1);
      ThisNumber := NumeralValue(ThisLetter);
      Denary := Denary + ThisNumber;
   END;
WriteLn(Denary);
```

[8]

(d) Mark as follows:

1 mark for per row

RomanNumber	Expected result	Reason for choice
"MDCLXVI"	1666	Each letter used once in descending order
"CCC"	300	Multiple letters (but not 4 identical letters)
"IIII"	4	Multiple letters (4 identical letters)
"IV"	4	Lower value letter followed by higher value letter
"XIV"	14	Order of letters: higher value, lower value, higher value
"XY"	Error (Do not accept –1)	Invalid symbol Y // invalid data

[4]

(e) (i)

RomanNumber	ThisLetter	ThisNumber	i	NextLetter	NextNumber	Denary
"IV"	'I'	1				0
		-1	2	'V'	5	-1
		5				4

1 mark for each row above (accept rows spread over more than one row) [2]

(ii)

RomanNumber	ThisLetter	ThisNumber	i	NextLetter	NextNumber	Denary
"XY"	'X'	10				0
			2	'Y'	-1	10
		-1				9

1 mark for each row above [2]

(iii) – does not give expected result // logic error

Change required:

– if value returned from `NumeralValue` function is – 1
– need error trapping code // error message [3]

(f) **(i)** – during compilation of program // in IDE environment // running an interpreted program
– reported by the translator diagnostics
// highlights/stops at the statement with the syntax error
// compiler checks against syntax rules/rules of the language

Accept by example [2]

(ii) – during testing (running code is not enough)
– When (using test data and) expected results do not match actual results [2]

3 **(a)** *Also give credit for answers to "why" rather than "how"*

(i) Set a breakpoint in the program code
Execution will stop at this point [2]

(ii) Stepping allows one statement to be executed at a time
Program execution pauses after each statement
Often used from a set break point
Can use variable watch at each step
Stepping over to skip statements [max 2]

(iii) Variable watch allows tester to see the current contents of a variable
// Used to see how variable contents change when stepping through program
Tester chooses variables to watch [2]

(b) White-box [1]