CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the October/November 2014 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 2014 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

1 (a) (i) *Mark as follows:*
1 mark for suitable labels/explanations for fields
1 mark for name and age entry options
1 mark for radio buttons or similar for Boolean club member field
1 mark for event choice (e.g. drop down list or radio buttons)
1 mark for fee box
1 mark for Confirm button [6]

(ii) Up to two marks for justification of features used in (i) [2]

(b) (i)

Field Name	Data Type	Field Size (bytes)
`CompetitorName`	String	26 (approx.) 15–40
`CompetitorAge`	Integer /Byte /ShortInt	4 1 2
`ClubMember`	Boolean	1
`EventEntered`	Char/Character	1/2
`EntryFee`	Currency/Real/float/single /decimal	4/8 /16

1 mark for each cell correct (Do not give a mark for a range) [10]

(ii) *1 mark for adding all 5 field lengths together (e.g. 40 bytes)*
1 mark for multiplying by 100 (e.g. 4000 bytes)
1 mark for adding 10% overheads (e.g. 4400 bytes) [3]

2 (a)

Loop	1	2	3	4	5	6	7	8
	s	z	x	y	m	List[m]	List[m] = s	List[m] > s
	64	–	1	15	–	–	–	–
		FALSE			–	–	–	–
1			**9**		8	52	**FALSE**	**FALSE**
2				**11**	**12**	**79**	**(FALSE)**	**TRUE**
3				**9**	**10**	**67**	**(FALSE)**	**(TRUE)**
4		**TRUE**			**9**	**64**	**TRUE**	

OUTPUT 9 [8]

1 mark for each column 2 to 8 correct (if no marks mark row by row)
1 mark for OUTPUT correct

(b) – searches for s (64) // (binary) search
– outputs position/index of requested value in list [2]

3 (a) (i) 1 mark for suitable values for white and black tokens
1 mark for suitable value for empty cell (e.g. NULL, "", 0, –1)

(ii) e.g. Pascal

```
VAR Grid : Array[1..6, 1..7] OF CHAR;        // 3 marks
FOR Row := 1 TO 6 DO                         // 1 mark
   FOR Column := 1 TO 7 DO                   // 1 mark
      Grid[Row, Column] := NULL;             // 2 marks
```

Mark as follows:
1 mark for correct identifier
1 mark for correct dimensions (6 × 7 or 7 × 6 elements)
1 mark for data type (needs to match the assignment)
1 mark for outer loop
1 mark for inner loop
1 mark for correct indexes
1 mark for correct assignment of a value to represent an empty cell

No marks for pseudocode [7]

(iii)
```
Grid[2, 4] := 'X';                           // 2 marks
```
[2]

(b) e.g. Pascal

```
FOR Row := 6 DOWNTO 1 DO
   BEGIN
      FOR Column := 1 TO 7 DO
         Write(Grid[Row, Column]);
      Writeln;
   END;
```

1 mark for correctly counting down
1 mark for correctly nested loops
1 mark for correct output statement with correct array element indexes
1 mark for correct new line (i.e. new line in outer loop only)
1 mark for appropriate indentation and suggested variable names (row, column, grid) [max 4]

(c) (i)

```
FUNCTION ColumnNumberValid(x : INTEGER) RETURNS BOOLEAN
    DECLARE Valid : BOOLEAN
    IF (x < 1) OR (x > 7)  // x outside range?
       THEN
          Valid ← FALSE          // column number not within range
       ELSE
          IF Grid[6, x] = NULL   // cell in top row empty?
             THEN
                Valid ← TRUE     // cell empty
             ELSE
                Valid ← FALSE    // cell not empty
          ENDIF
    ENDIF
    RETURN Valid
ENDFUNCTION
```

1 mark for each gap correctly filled [8]

(c) (ii)

Type of test data	Example test data	Justification
Normal/valid	Any integer between 1 and 7	A column number with top row free
Boundary/Borderline	Any integer between 1 and 7	A column number with column full/nearly full Accept boundary values for column number, e.g. 1/7 (first or last column)
Erroneous/Invalid	Any integer out of range (<1 or >7)	out of range

1 mark per cell correctly entered [9]

(d)
```
01 REPEAT
02     INPUT ChosenColumnNumber
03 UNTIL ColumnNumberValid(ChosenColumnNumber)
04 Row ← 1  // start with bottom row and find first empty row
05 WHILE Grid[Row, ChosenColumnNumber] <> NULL
06     Row ← Row + 1
07 ENDWHILE
08 IF NextPlayer = 'A'
09     THEN
10         Grid[Row, ChosenColumnNumber] ← 'O' // 'X'
11     ELSE
12         Grid[Row, ChosenColumnNumber] ← 'X' // 'O'
13 ENDIF
```

1 mark each for completing lines 3, 5, 6, 8.
1 mark for completing lines 10 and 12 correctly [5]

(e) (i) Player: passed by value 1 mark
Number: passed by reference 1 mark [2]

(ii)
```
GetColumn(NextPlayer, ChosenColumnNumber)
```

1 mark for each correct parameter [2]

(f) – indentation
– meaningful identifiers
– Initialising variables
– annotation/comments
– parameters
– procedure calls/modular structure
– keywords in capital letters [max 3]