CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the May/June 2013 series

9691 COMPUTING

9691/23   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 2013 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

1 (a)

Data	Identifier	Data Type	Size (in bytes)
subject	Subject	string	**10-30 (single value only) } 1**
examination title	**Suitable identifier 1**	**string/text 1**	**10-40 (single value only) }**
level	Level	**char 1**	1
date sat	DateSat	**string/text/date } 1**	**4/6/8/10 } 1**
mark	Mark	**Integer }**	3 }

[6]

(b) – addition of their field sizes
– add 10% (x)
– multiply 5 by 1024
– divide by their (x)   [4]

(c) (i) – ExamID / comparable
– integer/ other suitable   [2]

(ii) e.g. Pascal

```
Type Exam = RECORD
              Subject : String [10] ;
              Title:  String [20] ;
              Level: Char ;
               DateSat ; String [8] ;
                Mark ; Integer ;
         END ;
```

e.g. VB 2005

```
STRUCTURE Exam
        DIM Subject AS String
        DIM Title AS String
        DIM Level AS String
        DIM DateSat AS Date
        DIM  Mark AS Integer
END STRUCTURE
```

– Correct record header
– Definition terminator
– Date declared correctly
– All other fields declared correctly   [4]

(d) – easier to follow logic of problem
– can focus on one part at a time
– produces reusable code
– easier to maintain
– can debug a small section at a time   [Max 4]

(e) – uses/detects a marker written to the file …
– … immediately after the last record
– when processing a variable length file
– records can be processed until the marker is reached
– returns a Boolean value [Max 2]

2 (a) (Mark >= 0) AND (Mark <= 100)
AND
1st condition
2nd condition [3]

(b) (i)

Count	Mark	Mark>70	Output
1			
	28		
		False	
	57		
		False	
	75		
		True	
2			
	41		
		False	
			2

Each column 1 mark [4]

(ii) Count <- 0 [1]

(iii) Logic(al) error [1]

(iv) Increments the count of the number of exams with a mark over 70 [1]

(c)
```
      Count ← 0
WHILE NOT EOF()
  FILEREAD next assignment record
  IF Mark > 70
  THEN
      COUNT ← Count + 1
  ENDIF
ENDWHILE
```

- Initialising Count
- WHILE NOT(EOF)
- ENDWHILE
- IF Mark > 70   block   [4]

3 (a) – at the beginning / before any modules   [1]

(b) – difficult to find where variable value was changed
- makes re-use of modules more difficult
- two threads running simultaneously could try to modify the value   [Max 1]

(c) Integer value outside range 0–100/ null value   [1]

(d) e.g. VB 2005

```
DIM MyMarks(50) AS INTEGER
DIM Count AS INTEGER
FOR Count = 1 TO 50
    MyMarks(Count) = -1
NEXT Count
```

C#
```
Int [ ] myMarks;
myMarks = new int[50];
for (int count =1; count<50; count++)
   {
   myMark [count] = -1;
   }
```

- array declaration
- FOR loop
- assigning each element their value from (c)
- Loop ending   [4]

(e) e.g. VB 2005

```
Highest = 0
Lowest = 0
FOR Count = 1 TO 20
      IF MyMarks (Count) > Highest THEN
              Highest = MyMarks (Count)
      END IF
      IF MyMarks(Count) < Lowest THEN
              Lowest =MyMarks(Count);
      END IF
Console.WriteLine("Highest: ", Highest)
Console.WriteLine("Lowest: ",Lowest)
```

– FOR loop
– Setting a low highest value
– Setting a high lowest value
– Comparing each element with both these
– getting correct highest and lowest values
– output of values   [6]

(f) – ROUND(int)/ INT(var = 0.5)   [1]

(g) (i) Procedure returns 0, 1 or many values, function always returns 1 value   [1]

(ii) – One value, that of AvMark is required to be returned
– Either a function or a procedure could do that   [2]

4 (a) – sound output
– voice recognition
– facility to enlarge characters
– facility to change font
– facility to change colours
– less information on any one screen   [Max 3]

(b) – clear places for data entry
– button/method to change font sizes
– button/method to set sound input
– button/method to start sound synthesis
– method of changing colours
– simple screen layout
– buttons/method for moving between screens
– title   [Max 6]

(c) – logic error
Such as - wrong structure in an expression/variable not initialised/statement in wrong block
– run-time error
Such as – division by zero/using an array element that doesn't exist   [4]

5
```
       Count ← 0
       For i = 1 TO 20
              BEGIN
                 THEN OUTPUT 'Failed'
                   IF MyMarks[i] < 40
                     ELSE
                        IF MyMarks [i] >70
                           THEN
                        OUTPUT 'Distinction'
                        Count ← Count + 1
                     END IF
              END IF
       IF Count >3
          Then OUTPUT 'Well Done'
         END IF
```

- Initialising number of distinction marks
- loop to work through values
- test < 40
- Correct output
- test > 70
- correct output
- end of loop
- distinction total > 3
- IF…. END IFs match   [9]