CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the October/November 2013 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]. 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 2013 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

CAMBRIDGE
International Examinations

1 **(a)** – easier to understand the problem
– when dealing with smaller problems
– simpler to debug
– small parts at a time
– easier to maintain
– changing small sections
– can re-use his modules
– in this and future work

or comparable
3 × 2 marks [6]

(b) (i) – structure diagram/Jackson diagram/comparable
– orders sequence of modules/comparable [2]

(ii) – top level
– 3rd level in order [2]

(c) – procedures
– functions
– block structures [max 2]

(d) – using parameters
– to pass information about a data item
– can be by value or by reference [max 2]

(e) – array fixed size
– file not fixed
– array data volatile
– file saved
– array can be multi-dimensional
– array data can be added to a specific location
– array direct access
– file sequential access [6]

(f) – array declaration and size
– type integer
– loop
– set each element to sensible value (probably 0) [4]

2 (a) (i) – each condition not separate
– IF(Index > 100) OR(index < 0) THEN OUTPUT “Error” [2]

(ii) – arithmetic must be on RHS
– NumberOfCopies[Index] = NumberOfCopies[Index] + 1 [2]

(iii) – assigning value of wrong type
– NumberOfCopies[Index] = 3
3 × 2 marks [2]

(b)
```
IF ResourceID < 1001
    THEN
       OUTPUT "Cabinet 1"
    ELSE
       IF ResourceID <=3000
          THEN
             OUTPUT "Cabinet 2"
                IF ResourceID MOD 2 = 0
                   THEN
                      OUTPUT "Drawer 1"
                   ELSE
                      OUTPUT "Drawer 2"
                ENDIF
          ELSE
             IF ResourceID <=5000
                THEN
                   OUTPUT "Cabinet 3"
                ELSE
                   OUTPUT "Invalid ID"
             ENDIF
       ENDIF
ENDIF
```

- 1st condition
- correct output including OUTPUT/PRINT or equivalent
- dealing with inner nesting of odd/even
- correct 2nd and 3rd conditions
- correctly nested
- indentation [6]

(c)
```
e.g. PASCAL
var  ResourceID : integer;
begin
    readln(ResourceID);
    case ResourceID of
        1..1000:        writeln('Cabinet 1');
        1001..3000: if ResourceID  mod 2 =  0 then
                            writeln('Cabinet 2, Drawer 1')
                        else
                            writeln('Cabinet 2, Drawer 2');
        3001..5000: writeln('Cabinet 3');
    else
            writeln('Invalid Resource ID');
    end;
end.
```

- CASE/SELECT header
- correct form of each case (no =)
- dealing with inner options (odd/even)
- output of resource allocation
- correct logic overall
- terminating statement [6]

3 **(a)** – prompt to enter name
– drop-down list for type
– drop-down calendar for date
– box for resource ID
– box for keeping place
– buttons/similar for other actions
– uses space available [max 6]

(b) – title and date
– heading for resource
– heading for keeping place
– method of grouping
– use of all page [5]

(c) – indentation
– meaningful variable names
– comments
– capitalisation of keywords/variables [4]

(d)
```
Flag ← 0
INPUT P
X ←  1
REPEAT
   X ← X + 1
          IF myresources [X] = P
             THEN
                OUTPUT keptin [X]
                Flag ← 1
             ENDIF
UNTIL flag = 1 OR X=5000
IF flag = 0
   THEN
      OUTPUT "Not Found"
ENDIF
```

mark as follows
– indentation
– comments
– initialising X
– incrementing X
– correct UNTIL condition
– correctly adopt code in bold (do not give if FOR loop left in) [6]

(e) **(i)** – logic error ONLY
– their example (must make sense)
– run-time/syntax/semantic/logic (if not above) error
– their example [4]

(ii) – break point is a point where the program can be halted
– to see the current values of variable
– memory locations and registers
– stepping looks at one statement at a time
– to see the effect of each instruction [max 3]

4 **(a)** – within the function [1]

(b) – 7 [1]

(c) – Add(2)
– Add(1)
– Add(0) [3]