CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the May/June 2013 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 May/June 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 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 2]

(b) (i) –courseworkID/other comparable
–integer/other sensible [2]

(ii) PASCAL

```
TYPE Assignment = RECORD
                      CourseworkID : String[6];
                      Subject : String[10];
                      Title : String[10];
                      DateSet : TDateTime;
                      HandInDate : TDateTime;
                      IsMarked : Boolean;
                      DateReturned : TDateTime;
                      Mark : Integer;
                  END;
```

VB.NET / VB2005

```
STRUCTURE Assignment
    DIM CourseworkID AS String
    DIM Subject AS String
    DIM Title AS String
    DIM DateSet AS Date
    DIM HandInDate AS Date
    DIM IsMarked AS Boolean
    DIM DateReturned AS Date
    DIM Mark AS Integer
END STRUCTURE
```

VB6

```
Type Assignment
    CourseworkID AS String * 6
    Subject AS String * 10
    Title AS String * 10
    DateSet AS Date
    HandInDate AS Date
    IsMarked AS Boolean
    DateReturned AS Date
    Mark AS Integer
End Type
```

Note: string lengths optional

PYTHON

```
class Assignment :
    CourseworkID = "";
    Subject = ""
    Title = ""
    DateSet = datetime.date(1,1,1)
    HandInDate = datetime.date(1,1,1)
    IsMarked = False
    DateReturned = datetime.date(1,1,1)
    Mark = 0
```

Marking guidelines:
1 mark for correct record header
1 mark for correct definition terminator
1 mark for all 3 dates declared correctly
- *DateSet*
- *HandInDate*
- *DateReturned*

1 mark for the following fields defined correctly for language
- *Subject*
- *Title*
- *IsMarked*
- *Mark* [4]

(iii) 1 [1]

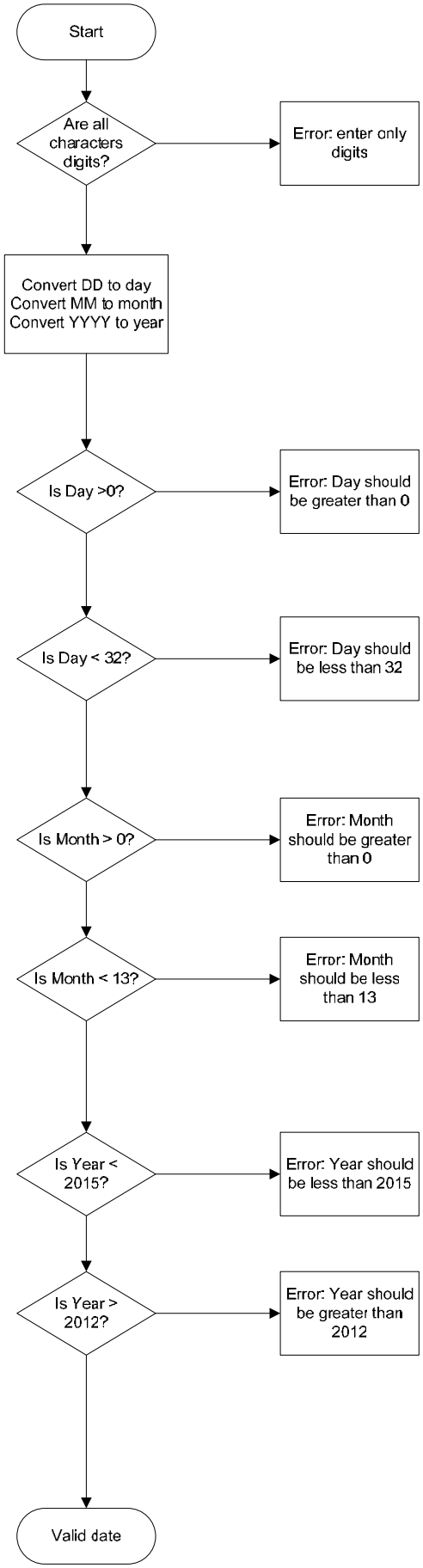
(c) –uses/detect a marker written to the file …
–… immediately after the last record
–when processing a variable length file
–records can be processed until the marker is reached
–returns a Boolean value [Max 2]

(d)
```
Found ← FALSE
WHILE NOT EOF(MyAssignments) AND NOT FOUND DO
  Read next Record
  IF Assignment.Subject = "Physics"
     THEN
       Found ← TRUE
  ENDIF
ENDWHILE;
```

Marking guidelines:
–set record found to false
–while NOT EOF and record found is false
–read next record
–check subject field to see if it is the wanted one
–if it is, set record found to true [Max 4]

2 (a) (IsMarked = 'Y') OR (IsMarked = 'N')
1 mark for OR
1 mark for the expressions (accept without brackets) [2]

(b)



Marking guidelines:
1 mark for checking that all characters are digits
1 mark for checking string length
1 mark for process of extracting substrings DD, MM, YYYY
1 mark for checking DD (1-31)
1 mark for checking MM (1-12)
1 mark for checking YYYY (2013-2014)
1 mark for "invalid date"
1 mark for "valid date" [Max 5]

(c) (i) 31122014 – borderline
16062013 – normal [1]

(ii) –you cannot tell which of the three components in invalid [1]

(iii) *Marking guidelines:*
1 mark for one with invalid DD only
1 mark for one with invalid MM only
1 mark for one with invalid YYYY only [3]

(d) (HandInDate > DateSet) AND (HandInDate > CurrentDate)

1 mark for AND
1 mark for correct expressions (accept without brackets) [2]

(e)
```
Valid ← FALSE
IF DateReturned >HandInDate
  THEN
    IF DateReturned <= CurrentDate
      THEN
        IF (Mark >= 0) AND (Mark <= 100)
          THEN Valid ← TRUE
        ENDIF
    ENDIF
ENDIF
```

Marking guidelines:
1 mark for nested IFs or ELSEIFs
1 mark for correct number of ENDIF(s)
1 mark for DateReturned>HandInDate
and DateReturned<=CurrentDate
1 mark for process mark checked
1 mark for "valid" and "invalid" correctly reported or assigned [Max 4]

(f) (i)

Count	Mark	Mark<40	Output
0			
	28		
		True	
1			
	55		
		False	
	70		
		False	
	12		
		True	
2			
			2

1 mark for each column [4]

(ii) –gives the number of assignments with a mark less than 40/failed [1]

(iii) –indentation
–sensible variable names
–keywords in capitals [2]

(iv) –comments/annotation [1]

(v) –any pseudocode example with a useful comment [1]

(vi)
```
Count ← 0
WHILE NOT EOF()
   FILEREAD next assignment record
   IF Mark < 40
      THEN
         COUNT ← Count + 1
   ENDIF
ENDWHILE
```

1 mark for initialising count & FILEREAD & IF statement correctly copied (bold code above)
1 mark for WHILE NOT EOF( ) in correct place
1 mark for ENDWHILE in correct place [3]

(vii) –no
–don't know length of file/how many records [2]

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) –within the module/subroutine/block in which it is declared [1]

(d) a suitable number, e.g. –1 (not any value between 0 and 100 inclusive)
reason: this mark is a dummy/rogue value [1]

(e) **PASCAL**

```
VAR  Marks : ARRAY[1..30] OF INTEGER;
 i : INTEGER;

FOR i := 1 to 30 DO
BEGIN
 Marks[i] := -1;
END;
```

VB.NET / VB2005

```
Dim Marks(30) AS Integer
DIM i AS Integer

For i = 1 to 30
     Marks(i) = -1
NEXT i
```

VB6

```
DIM Marks(30) AS INTEGER
DIM i AS Integer

FOR i = 1 TO 30
      Marks(i) = -1
NEXT i
```

PYTHON

```
Marks = []

for i in range(0, 30) :
    Marks.append(-1)
```

Marking guidelines:
1 mark for correct array declaration
1 mark for correct FOR loop
1 mark for assigning the value given in (d) to each element
1 mark for LOOPEND / declaration end [4]

(f) PASCAL

```
VAR  Marks : ARRAY[1..30] OF INTEGER;
 AvMark : REAL;
 Count, Total, i : INTEGER;

BEGIN
 . . . . . .
     Total := 0;
     Count := 0;
 FOR i := 1 to 30 DO
 BEGIN
      IF Marks[i] > -1 THEN
      BEGIN
           Count := Count + 1;
           Total := Total + Marks[i];
      END;
 END;
 AvMark := Total/Count;
END.
```

VB.NET / VB2005

```
Dim Marks(30) AS Integer
. . . . . .
Dim Count AS Integer
Dim Total AS Integer
Dim i AS Integer
Dim AvMark AS Double

Total = 0
Count = 0
For i = 1 To 30 Then
 If Marks(i) > -1
      Count = Count + 1
      Total = Total + Marks(i)
 End IF
Next i
AvMark = Total/Count
```

VB6

```
DIM Marks(30) AS INTEGER
. . . . . .
DIM Count AS INTEGER
DIM Total AS INTEGER
DIM i AS INTEGER
DIM AvMark AS DOUBLE

Total = 0
Count = 0
FOR i = 1 TO 30
 IF Marks(i) > -1 THEN
       Count = Count + 1
       Total = Total + Marks(i)
 END IF
NEXT i
AvMark = Total/Count
```

PYTHON

```
Marks = []
. . . . . .
Total = 0
Count = 0
AvMark = 0

for i in range(0, 30) :
 if Marks[i] > -1 :
       Count = Count + 1
       Total = Total + Marks[i]
AvMark = Total/Count
```

Marking guidelines:
1 mark for initialisation of total and marks count
1 mark for fully functioning loop
1 mark for ignoring the elements with the initial value
1 mark for incrementing count correctly
1 mark for totalling and dividing and assigning the result to AvMark [5]

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

(ii) It returns one value, AvMark [1]

(h) (i) 34 [1]

(ii) 80 [1]

(iii) PASCAL

```
FUNCTION CalculateRounded(AvMark : REAL) : INTEGER;
VAR Rounded : INTEGER;
BEGIN
       Rounded := TRUNC(AvMark + 0.5);
```

```
        CalculateRounded := Rounded;
END;
```

VB.NET / VB2005

```
Function CalculateRounded(ByVal AvMark AS Double) AS
Integer
       Dim Rounded As Integer
       Rounded = INT(AvMark + 0.5)
       CalculateRounded = Rounded // or: Return Rounded
End Function
```

VB6

```
Function CalculateRounded(AvMark AS Double) AS Integer
     Dim Rounded AS Integer
     Rounded = INT(AvMark + 0.5)
     CalculateRounded = Rounded
End Function
```

Note: data type optional for parameter

PYTHON

```
def CalculateRounded(AvMark) :
     Rounded = int(AvMark + 0.5)
     return Rounded
```

Marking guidelines:
1 mark for function heading including return data type if applicable
1 mark for parameter including data type if applicable
1 mark for calculation
1 mark returning value [Max 4]

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) (i) When:
–during compilation/interpretation/translation//while code is written in an IDE
How:
–the compiler/interpreter/IDE checks that the rules of the language are being followed [2]

(ii) When:
–when unexpected results occur
How:
–dryrun/trace/white-box/debugging [2]