UNIVERSITY OF CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the October/November 2011 question paper for the guidance of teachers

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.

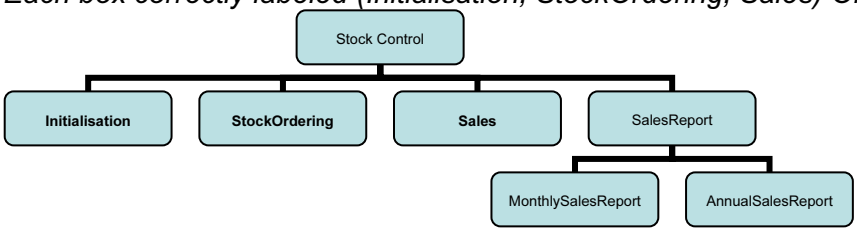
Mark schemes must be read in conjunction with the question papers and the report on the examination.

- Cambridge will not enter into discussions or correspondence in connection with these mark schemes.

Cambridge is publishing the mark schemes for the October/November 2011 question papers for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level syllabuses and some Ordinary Level syllabuses.

1 **(a)** e.g.
-each can work on individual modules
-modules can be written in parallel
(answer must be specific to this scenario) Max [ ]

(b) *Each box correctly labeled (Initialisation, StockOrdering, Sales) Order significant*



[1]

(c) *1 mark for 2 boxes under SalesReport*
1 mark for correct labelling [2]

(d) -these will be local variables
-that only have effect in the module they are in // local scope
-stored in different memory locations
-and have no meaning outside that module Max [2]

(e) **(i)** -keywords/reserved words
-a word in the vocabulary of the language
-that can only have the meaning defined in that language Max [1]

(ii) e.g. Visual Basic:
-names must begin with a letter
-must not contain a space/punctuation characters/certain characters
-must be unique in their block/scope
-can't be more than 64 characters
-can't be a keyword Max [3]

(iii) Any keyword // word breaking a rule given by the candidate [1]

(f) **(i)** 604 [1]

(ii) (a+b)/100 [1]

(iii) Black box CAO [1]

(g) **(i)** -valid/normal data
-extreme / boundary data [2]

(ii) 6 different types of test data sets + 6 sensible reasons
Reason must relate to the scenario
Value + correct reason = 1 mark [6]

(h) (i) (PromotionCode="gold") OR (PromotionCode="silver") OR (PromotionCode="bronze")

1 mark for 3 separate correct conditions
1 mark for ORs

Alternative answer:
PromotionCode IN ["gold","silver","bronze"]
2 marks (1 mark for IN, 1 mark for ["gold","silver","bronze"]) [2]

(ii) -wrong or no promotion rate applied
-the program would not find associated records [2]

(iii) *1 mark for clear information*
1 mark for choice as a drop-down list
1 mark for move on button Max [2]

2 (a) (i) -Valid data entered CAO [1]

(ii) -Invalid data. Try again CAO [1]

(b) e.g. Pascal

```
READLN (Position);
IF Position = 'F'
  THEN WRITELN('Valid data entered')
  ELSE
    IF Position = 'D'
      THEN WRITELN('Valid data entered')
      ELSE
        IF Position = 'G'
          THEN WRITELN('Valid data entered')
          ELSE WRITELN('Invalid data. Try again');
```

e.g. VB6

```
Position = txtBox.Text
IF Position = "F" THEN
  MsgBox "Valid data entered"
ELSEIF Position = "D" THEN
  MsgBox "Valid data entered"
ELSEIF Position = "G" THEN
  MsgBox "Valid data entered"
ELSE
  MsgBox "Invalid data. Try again"
END IF
```

e.g. VB 2005

```
Position = Console.ReadLine
IF Position = "F" THEN
  Console.WriteLine("Valid data entered")
ELSEIF Position = "D" THEN
  Console.WriteLine("Valid data entered")
ELSEIF Position = "G" THEN
  Console.WriteLine("Valid data entered")
ELSE
  Console.WriteLine("Invalid data. Try again")
END IF
```

e.g. C#

```
position = Console.ReadLine();
if (position = "F")
  {
    Console.WriteLine("Valid data entered");
  }
else if (position = "D")
  {
    Console.WriteLine("Valid data entered");
  }
else if (position = "G")
  {
    Console.WriteLine("Valid data entered");
  }
else
  {
    Console.WriteLine("Invalid data. Try again");
  }
```

1 mark for correct input
1 mark for 1st condition correct
1 mark for all conditions correct
1 mark for correct output for valid input
1 mark for correct output for invalid input
1 mark for conditions indented Max [5]

(c) -Sequence, selection *(in any order, these words only)* [1]

(d) -A process of repeating
-A block of statements/number of steps
-Until some condition is met Max [2]

(e) *1 mark for a counter variable*
1 mark for correctly initialising counter
1 mark for incrementing counter
1 mark for correct condition for terminating
1 mark for correct output Max [5]

(f)

Field Name	Data Type	Field Size (bytes)	
PlayerID	Integer/byte/shortint	a value within 1–6	*NOT a range*
Sex	Boolean/character	1	
PlayerName	String/Text	a value within 10–50	*NOT a range*
Position	Character/String	1	
DateOfBirth	Date/Integer/String	2/4/6/8	

1 mark per cell [10]

(g) (i) -logic (error) [1]

	(i)	(ii)
EITHER:	Index ← 1	Index ← 0
OR:	UNTIL Index = 45	UNTIL Index >45 or UNTIL Index = 46
	[1]	[1]

(h)
```
Gtotal ← 0
FOR Index ← 1 TO 45
  IF Club[Index].Position = 'G'
    THEN
      Gtotal ← Gtotal + 1
  ENDIF
ENDFOR
```

1 mark for correct FOR loop
1 mark for correct content of IF statement and condition
1 mark for ENDFOR in correct position or equivalent structure [3]

3 (a)

s	x	q[1]	q[2]	q[3]	q[4]	Surprise
CHO JABA						
	1					
		C				
	2					
			H			
	3					
				O		
	4					
						CHO

1 mark for correct x values (2,3,4)
1 mark for correct q values (C, H, O)
1 mark for correct surprise (CHO) [3]

(b) -pick out the first word of a sentence/group of words [1]

(c) -assigns return value to Surprise
-that value is returned to the function call
-name of function used as a variable Max [2]

(d) -is a subroutine // can be called more than once // can be called from different locations
-given a name/identifier
-may take parameter values from the program
-returns value to the program Max [3]

(e) (i) -ends REPEAT
-by finding an empty space
-indicating end of word Max [2]

(ii) -indentation
-meaningful/sensible variable names [2]

(f) -characters are compared in turn
-from the left hand side/start of each word
-the first higher code value determines the largest word
-if 2 words are the same when one ends
-the other is the larger alphabetically Max [3]