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

9691/22 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 must be read in conjunction with the question papers and the report on the examination.

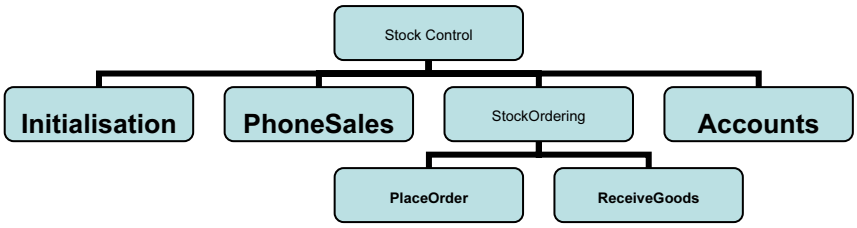
- Cambridge will not enter into discussions or correspondence in connection with these mark schemes.

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.

UNIVERSITY of CAMBRIDGE
International Examinations

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

(b) *Each box correctly labelled (Initialisation, PhoneSales, Accounts) Order significant*



[1]

(c) *1 mark for 2 boxes under StockOrdering*
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) Any keyword // word breaking their language's rule [1]

(iii) 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 [3]

(f) **(i)** 580 CAO [1]

(ii) (a-b)*0.1 [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) (ContractLength=12) OR (ContractLength=18) OR (ContractLength=24)

1 mark for 3 separate correct conditions
1 mark for ORs

Alternative answer:
ContractLength IN [12,18,24]
2 marks (1 mark for IN, 1 mark for [12,18,24]) [2]

(ii) (FreeTexts >= 0) AND (FreeTexts <= 600)

Alternative: (FreeTexts>-1) AND (FreeTexts < 601)

1 mark for 2 separate correct conditions
1 mark for AND

Alternative answer:
NOT ((FreeTexts < 0) OR (FreeTexts > 600)) [2]

(iii) -data outside expected range / invalid data could be entered
-the program would not find an associated record/data [2]

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

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

(b) e.g. Pascal

```
READLN (NumberOfYears);
CASE NumberOfYears OF
  0..2: WRITELN('Valid data entered');
  ELSE WRITELN('Invalid data. Try again');
END;
```

e.g. VB6

```
NumberOfYears = txtBox.Text
SELECT CASE NumberOfYears
  CASE 0 TO 2
    MsgBox "Valid data entered"
  CASE ELSE
    MsgBox "Invalid data. Try again"
END SELECT
```

e.g. VB 2005

```
NumberOfYears = Console.ReadLine
SELECT CASE NumberOfYears
  Case 0 TO 2
    Console.WriteLine("Valid data entered")
  Case Else
    Console.WriteLine("Invalid data. Try again")
END SELECT
```

e.g. C#

```
numberOfYears = Console.ReadLine();
switch (numberOfYears)
{
  case 0:
  case 1:
  case 2:
    Console.WriteLine("Valid data entered");
    break;
  default:
    Console.WriteLine("Invalid data. Try again");
    Break;
}
```

e.g. Java

```
numberOfYears = reader.readLine();
switch (numberOfYears) {
  case 0:
  case 1:
  case 2:
    System.out.println("Valid data entered");
    break;
  default:
    System.out.println("Invalid data. Try again");
    break;
}
```

Accept write.write *instead of* System.out.println

1 mark for correct input
1 mark for CASE statement correct
1 mark for all cases correct
1 mark for correct output for Valid data entered
1 mark for correct output for Invalid data. Try again
1 mark for conditions indented
No marks for pseudocode (give marks for clear code, even if unidentified)
For Python accept if … elif …. elif … else … 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 from decision [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*
NumberOfYears	Integer/byte/shortint	a value within 1–6	*NOT a range*
DateOfBirth	Date/Integer/String	2/4/6/8/10	

1 mark per cell [10]

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

	(i)	(ii)
EITHER	Index ← 1	Index ← 0
OR	WHILE Index < 30	WHILE Index <=30 or WHILE Index < 31
	[1]	[1]

(h)
```
NoOfFemales ← 0
FOR Index ← 1 TO 30
  IF Squad[Index].Sex = 'f'
    THEN
      NoOfFemales ← NoOfFemales + 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)

MyWord	i	LENGTH (MyWord)	TempNo	Temp Word[1]	Temp Word[2]	Temp Word[3]	Surprise
cab							
	1						
		3					
			100				
				d			
	2						
			98				
					b		
	3						
			99				
						c	**dbc**

1 mark for correct i (2,3)
1 mark for correct TempWord[1] (d)
1 mark for correct TempWord[2] and TempWord[3] (b and c)
1 mark for correct surprise (dbc) Max [3]

(b) -codes each letter to the next letter [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
-may return parameter values to the program Max [3]

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

(ii) -evaluates the ASCII value of the current character
-adds 1 to that value
-puts the new character in current letter position [3]

(f) -joining
-two strings together [2]