UNIVERSITY OF CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the May/June 2012 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 May/June 2012 question papers for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level syllabuses and some Ordinary Level syllabuses.

1 **(a)**
- labelled box for name
- drop-down list/radio buttons for age
- labelled space for sport
- yes/no radio buttons or (drop down) list
- button to move from screen [4]

(b)
- easy to use
- clear instructions
- appropriate for the purpose
- to reduce errors
- easy to understand [3]

(c)
- possibility of making fonts larger
- incorporating sound
- changing colours [2]

(d)

Field Name	**Data Type**	**Field Size (bytes)**
FirstName	String/alphanumeric/text	8–20 (single value only)
Age	Integer	2
FavouriteSport	String/alphanumeric/text	10–40 (single value only)
ClubMember	Boolean/char	1

[8]

(e) e.g.

```
Total16 ← 0
Total17 ← 0
Total18 ← 0
Total19 ← 0

REPEAT
    READ next record
    CASE age OF
        16: Total16 ← Total16 + 1
        17: Total16 ← Total16 + 1
        18: Total16 ← Total16 + 1
        19: Total16 ← Total16 + 1
    CASE ELSE
        Error
    ENDCASE
UNTIL no more student records
```

Marking guidelines:
- *sensible age total names*
- *initialising age totals (before given loop)*
- *sensible CASE/SELECT construct*
- *case sensibly formed*
- *end of CASE/SELECT* [5]

(f) (*File handling statement – 1 mark; explanation – 1 mark*) × 4

e.g. Pascal

```
AssignFile(Channel, ExternalFileName); -gives the FileNam
Channel ID through which access can be made
Reset(Channel); -opens existing file
Write(Channel, Record); -writes record to file
Read (Channel, Record); -reads record from file
Seek (Channel, RecordAddress); -goes directly to record at
specified address
CloseFile (Channel); -closes file
```

e.g. VB 2005

```
Channel = New FileStream(ExternalFileName,FileMode.Open)
FileReader = New BinaryReader(Channel)
NewFile = New FileStream(ExternalFileName, FileMode.Create
FileWriter = New BinaryWriter (NewFile)
Record.Field = FileReader.ReadString()
Record.Field = FileReader.ReadDecimal()
Record.Field = FileReader.ReadInt32()
FileWriter.Write(Field)
Channel.Close()
FileReader.Close()
FileWriter.Close()
NewFile.Close()
```

e.g. C#

```
channel = new FileStream(externalFileName,FileMode.Open)
fileReader = new BinaryReader(channel)
newFile = new FileStream(externalFileName, fileMode.Create
fileWriter = new BinaryWriter (newFile)
record.Field = FileReader.ReadString()
record.Field = FileReader.ReadDecimal()
record.Field = FileReader.ReadInt32()
fileWriter.Write(field)
channel.Close()
fileReader.Close()
fileWriter.Close()
newFile.Close()
```

[8]

2 **(a)** Method of marking:

- *sensible input string name*
- *Method of traversing string*
- *identifying **
- *method of deleting or avoiding **
- *checking > 1* at a place*
- *dealing with end of string*
- *sensible output string name*
- *works correctly when no *s*
- *indented code*

e.g. Pascal

```
ReadLn(OldString);
NewString := ‘’;
FOR i := 1 TO Length(OldString) DO
   IF OldString[i] <> ‘*’ THEN
      NewString := Concat(NewString,OldString[i]);
```

e.g. VB 2005

```
OldString = Console.ReadLine()
NewString = “”
FOR i = 1 TO OldString.Length
   IF OldString(i) <> “*” THEN
      NewString = String.Concat(NewString,OldString(i))
   END IF
NEXT
```

e.g. C#

```
oldString = Console.ReadLine();
newString = “”;
for (int i = 1;i <= oldString.Length; i++)
   {
   if (oldString[i] <> “*”)
      {
      newString = String.Concat(newString,oldString[i])
      }
   }
```

[8]

(b) **(i)** – OldString (their input string name) [1]

(ii) e.g. Pascal

```
Function ChangeString(OldString: String): String
```

e.g. VB 2005

```
Function ChangeString(ByVal OldString As String) As String
```

e.g. C#

```
static string changeString(string oldString)
```

[2]

(iii) – single output makes this appropriate [1]

3 (a)

RowNo	ColumnNo	RowNo<4	ColumnNo<4	Board				
				[1,1]	**[1,2]**	**[1,3]**	**[2,1]**	**[2,2]**
1	1	TRUE	TRUE	0				
	2				**0**			
	3					**0**		
	4		**FALSE**					
2	**1**		**TRUE**				**0**	
	2							**0**

1 mark for TRUE, TRUE for [1,2]
1 mark for [1,2] in top row
1 mark for TRUE, TRUE for [1,3] and entry in top row
1 mark for TRUE, FALSE for [1,4]
1 mark for TRUE, TRUE for next 2 entries
1 mark for [2,1] and [2,2] in top row [5]

(b) – Logic/logical [1]

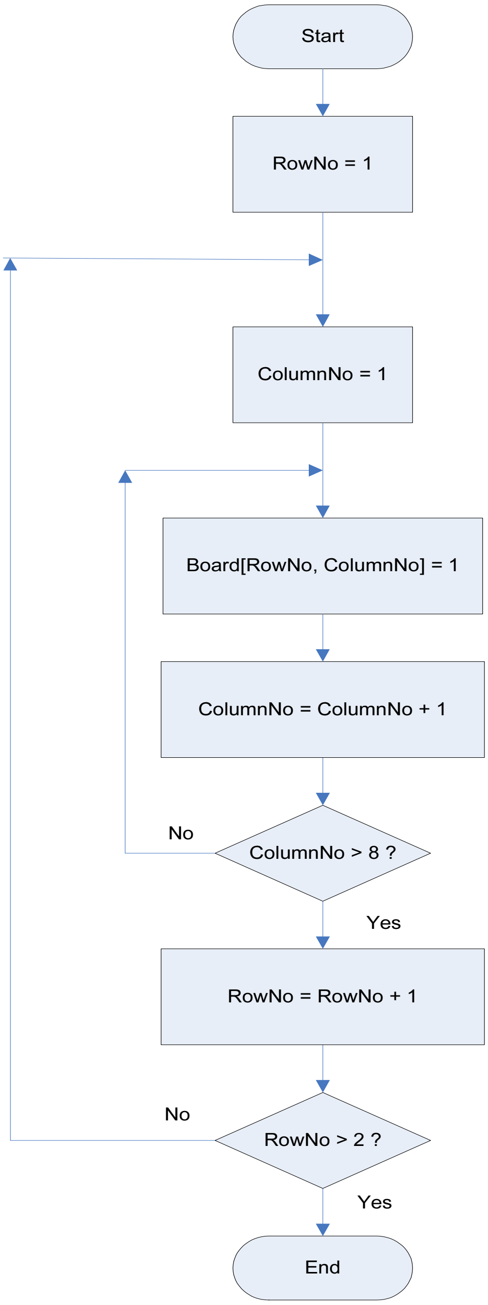
(c) – 2 WHILE RowNo <= 4 (or equivalent)
– 5 WHILE ColumnNo <= 4 (or equivalent) [2]

(d) – FOR loop [1]

(e) – at opposite corners [1]

(f) Marking guide:

–2 FOR loops
–nested
–row loop values only 1 & 2
–assigning value 1
–correct exit



[5]

4 **(a)** **(i)** 1.75 (or equivalent)

(ii) 3

(iii) 1 [3]

(b) **(i)** Y DIV X [1]

(ii) Y MOD X [1]

(c) DIV gives as answer the integer part
/gives the full division (integer and fractional part) [2]

5 **(a)** *There are many different ways to represent the working*
Example:

- Something(4) becomes Something(5) + 4
- Something(5) becomes Something(6) + 5
- Something(6) becomes Something(7) + 6
- Ends at 7
- Diagram works back through functions
- Something(4) = 16 [6]

(b) – infinite loop//runs out of stack space [1]

(c) – it would never end, whatever the input value
– … as no stopping condition [2]

(d) e.g.

```
Something ← 0
FOR x ← 4 TO 6
   Something ← Something + x
ENDFOR
Something ← Something + 1
```

Marking points:
–Initialising `Something`
–Correct `FOR` statement
–Adding 1 at the end [2]