CAMBRIDGE INTERNATIONAL EXAMINATIONS

GCE Advanced Subsidiary Level and GCE Advanced Level

MARK SCHEME for the October/November 2012 series

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.

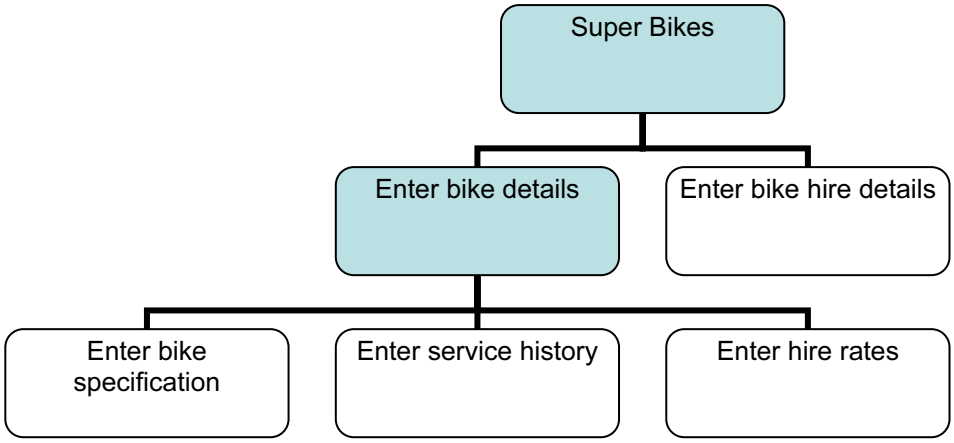
Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the October/November 2012 series for most IGCSE, GCE Advanced Level and Advanced Subsidiary Level components and some Ordinary Level components.

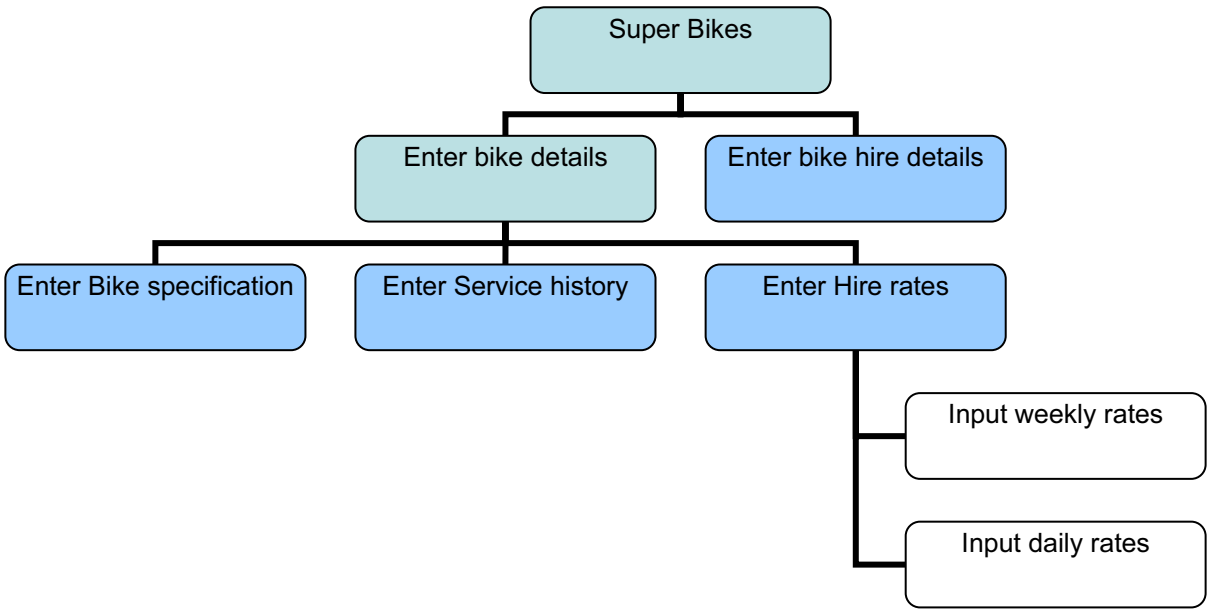
CAMBRIDGE
International Examinations

1 **(a)** Enter bike details



1 mark for correct order
1 mark for correct level [2]

(b)



1 mark for 2 blocks under Enter Hire Rates [1]

(c) **(i)** Invalid

(ii) Invalid

(iii) road is valid [3]

(d) IF (BikeType=”trial”) OR (BikeType=”scooter”) OR (BikeType=”road”) [1]

(e) (i) e.g. Pascal

```
1   VAR BikeRegValid : Boolean;
2   BikeRegValid := TRUE;
3   IF length(BikeReg) <> 5
4      THEN BikeRegValid := FALSE;
5   IF NOT((RIGHT(BikeReg,3)>='000')
6             AND (RIGHT(BikeReg,3)<='999'))
7      THEN BikeRegValid := FALSE;
8   IF LEFT(BikeReg,2) <> 'BK'
9      THEN BikeRegValid := FALSE;
10  IF BikeRegValid
11     THEN writeln('valid')
12     ELSE writeln('invalid');
```

e.g. VB 2005

```
1   BOOLEAN BikeRegValid
2   BikeRegValid = TRUE
3   IF LEN(BikeReg) <> 5 THEN
4      BikeRegValid = FALSE
5   END IF
6   IF NOT(MID(BikeReg,3,3)>="000"
7          AND MID(BikeReg,3,3)<="999") THEN
8      BikeRegValid = FALSE
9   END IF
10  IF MID(BikeReg,1,2) <> "BK" THEN
11     BikeRegValid = FALSE
12  END IF
13  IF BikeRegValid THEN
       Console.writeline("valid")
14  Else Console.writeline("invalid")
15  END IF
```

e.g. C#

```
1   bool bikeRegValid = true;
2   if (bikeReg.Length != 5)
3      {
4        bikeRegValid = false;
5      }
6   if (!((bikeReg.Substring(3,3)>="000")
7            && (bikeReg.Substring(3,3)<="999")))
8      {
9        bikeRegValid = false;
10     }
11  if (bikeReg.Substring(1,2) != "BK")
12     {
13       bikeRegValid = false;
14     }
15  if (bikeRegValid)
16     {
17       Console.WriteLine("valid");
18     }
19  else
20     {
21       Console.WriteLine("invalid");
22     }
```

e.g. Python

```
1     bikeReg = input()
2     bikeRegValid = True
3     if len(bikeReg) != 5:
4         bikeRegValid = False
5     if ((bikeReg[2:5] >='000') & (bikeReg[2:5] <= '999')) != True:
6         bikeRegValid = False
7     if bikeReg[0:2]!='BK':
8         bikeRegValid = False
9     if bikeRegValid:
10        print ('valid')
11    else:
12        print ('invalid')
```

1 mark for checking length is 5 characters
1 mark for correct separating 1st two characters
1 mark for testing first two characters are BK
1 mark for separating last three characters
1 mark for testing last three characters are digits
1 mark for initialising Boolean value
1 mark for changing Boolean value if error
1 mark for suitable message
1 mark for meaningful variable names used
1 mark for indentation [10]

(ii) – digits first 3 characters, rather than last 3 characters
– in above example at line numbers 5/6 (Pascal), 6/7 (VB, C#) [2]

(f) (i) – regards a block of software as an entity or black box
– tests inputs
– give correct outputs [3]

(ii) – takes every possible path
– through the block of code
– done by a programmer [3]

2 (a)

	Position	Row<=30	**Position <=4**	**BikeReg <>"BBB"**	BikeSpace				
					[1,1]	**[1,2]**	**[1,3]**	**[1,4]**	**[2,1]**
1	1	TRUE	**TRUE**	**TRUE**	BK707				
	2					**BK380**			
	3						**BK162**		
	4							**BK747**	
	5		**FALSE**						
2	**1**		**TRUE**						**BK913**

1 mark for second decision in heading
1 mark for third decision in heading
1 mark for correct array elements in heading
1 mark for correct values into array elements
1 mark for correct values in first 5 columns
1 mark for correct placing of FALSE [6]

(b) e.g. Pascal

```
Row := 1;
REPEAT
   Position := 1;
   REPEAT
      READLN(BikeReg);
      IF BikeReg = 'BBB' THEN Exit;
      BikeSpace[Row,Position] := BikeReg;
      Position := Position + 1;
   UNTIL Position > 4;
   Row := Row + 1;
UNTIL Row > 30;
```

e.g. VB 2005

```
Row = 1
DO
   Position = 1
   DO
      BikeReg = CONSOLE.READLINE()
      IF BikeReg = "BBB" THEN EXIT
      BikeSpace(Row,Position) = BikeReg
      Position = Position + 1    ' Position += 1
   LOOP UNTIL Position > 4
   Row = Row + 1       ' Row += 1
LOOP UNTIL Row > 30
```

e.g. C#

```
row := 1;
do
{
   position := 1;
   do
   {
       bikeReg = Console.ReadLine();
       if (bikeReg = "BBB")
       {
         exit;
       }
       bikeSpace[row,position] = bikeReg;
       position := position + 1;//position += 1;
   }
   WHILE (position <= 4)
   row := row + 1;  // row += 1;
}
WHILE (row <= 30)
```

1 mark for correct repeat loops
1 mark for correctly nested loops
1 mark for input in correct place
1 mark for correct incrementation
1 mark for checking for rogue value
1 mark for assignment to correct array element
1 mark for indentation [7]

(c) (i) 0 (zero) [1]

(ii) Run-time error (Allow logic error, arithmetic error) [1]

(iii) – check the value of the bracket before the division takes place // write error trapping code
– if bracket = 0 arrange for a message to be output // exception code [2]
Accept answers in code

(d) – set breakpoint at the beginning of the code under scrutiny
– select the variables in the watch window …
– … whose values need checking
– … while stepping through the program code
– a line at a time [4]

3 – date (month alone sufficient)
–suitable report title
– company name
– tabulated or other suitable layout
– grouped by insurance rating
– total income for each group shown
– well spaced out (making use of whole frame)
(if clearly a screen design do not give this mark) [7]

4 (a)

	Data Type	Size of Field (bytes)
BikeReg	String/alphanumeric/text	5
PurchaseCost	Currency/integer/real/decimal	8 (accept 4–8)
InsuranceRating	Char	1
ServiceDue	Boolean	1

[4]

(b) (5 + 8 + 1 + 1)
* 1000 / 1024
* 1.1 (or equivalent)
= 16.1KB (f.t.) [4]

(c) (i) e.g. Pascal

```
TYPE HireBike = RECORD
                  BikeReg: String[5];
                  PurchaseCost: Currency;
                  InsuranceRating: Char;
                  ServiceDue: Boolean;
                END;
```

e.g. VB 2005

```
STRUCTURE HireBike
   DIM BikeReg AS String
   DIM PurchaseCost AS Decimal
   DIM InsuranceRating AS Char
   DIM ServiceDue AS Boolean
END STRUCTURE
```

e.g. C#

```
struct hireBike
   {
     public string bikeReg;
     public decimal purchaseCost;
     public char insuranceRating;
     public bool serviceDue;
   }
```

1 mark for correct record heading
1 mark for correct record structure ending
1 mark for first 2 fields
1 mark each for 3rd and 4th field [5]

(ii) e.g. Pascal

```
ASSIGNFILE(FS, 'SuperBikes');
RESET(FS);
BikesForService := 0;
WHILE NOT EOF(FS) DO
BEGIN
   Read(FS, HireBike);
   IF HireBike.ServiceDue
      THEN
         BikesForService:=BikesForService+1;
END;
WRITELN('Number of bikes for service: ',
                                BikesForService);
CloseFile(FS);
```

e.g. VB 2005

```
FS = NEW FileStream("SuperBikes",
                                   FileMode.open)
BR = NEW BinaryReader(FS)
BikesForService = 0
DO WHILE FS.Position < FS.Length
   HireBike.ServiceDue =
                       BR.ReadBoolean()
   IF HireBike.ServiceDue THEN
      BikesForService = BikesForService + 1
LOOP
WRITELINE("Number of bikes for service: ",
                                BikesForService);
BR.Close()
FS.Close()
```

e.g. C#

```
fs = new Filestream("SuperBikes", FileMode.Open);
br = new BinaryReader(fs);
bikesForService = 0;
do
{
   hireBike.ServiceDue =br.Readbool();
   IF (hireBike.ServiceDue)
     {
         bikesForService:=bikesForService+1;
     }
}
while (fs.Position < fs.Length);
Console.Writeline("Number of bikes
                  for service: ",bikesForService);
br.Close();
fs.Close;
```

1 mark for initialising total
1 mark for assigning file name
1 mark for opening file for reading
1 mark for repeat/while loop
1 mark for reading record
1 mark for testing 'service due' field set to true
1 mark for incrementing total
1 mark for outputting total
1 mark for closing file [9]