CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Level

MARK SCHEME for the October/November 2015 series

9608 COMPUTER SCIENCE

9608/42 Paper 4 (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.

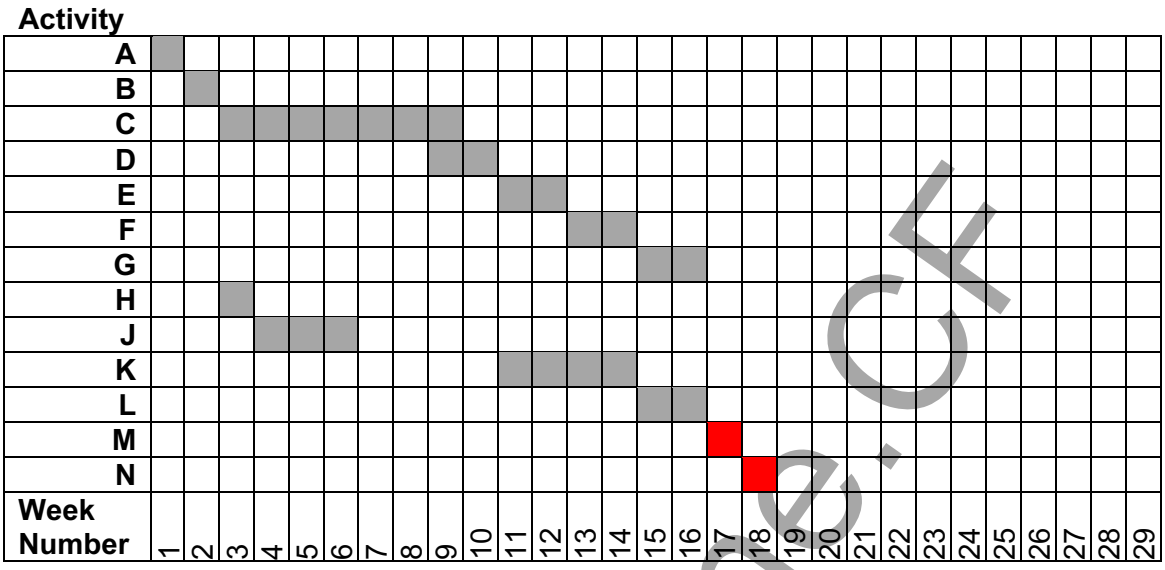
Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the October/November 2015 series for most Cambridge IGCSE®, Cambridge International A and AS Level components and some Cambridge O Level components.

CAMBRIDGE
International Examinations

1 (a) (i)



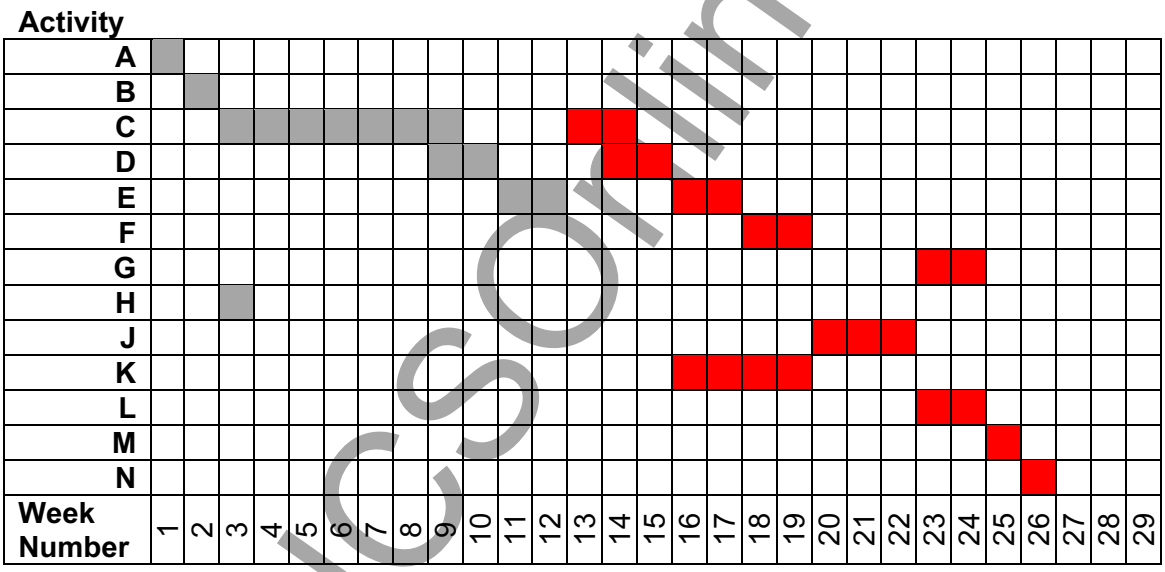
1 mark for each square [2]

(ii) week number 18

Allow follow through [1]

(b) (i)

Activity	Description	Weeks to complete
A	Write requirement specification	1
B	Produce program design	1
C	Write module code	7
D	Module testing	2
E	Integration testing	2
F	Alpha testing	2
G	Install software and carry out acceptance testing	2
H	Research and order hardware	1
J	Install delivered hardware	3
K	Write technical documentation	4
L	Write user training guide	2
M	Train users on installed hardware and software	1
N	Sign off final system	1



1 mark per activity (but 1 mark for activity M and N)
Notes:
C must be after E (1 or 2 later is ok)
D, E, F correct relative to C
J must start in week 20 (allow 21, 22)
G must come after the end of J (f.t.)
K finishes after or at same time as F
L finishes at the same time as G **and after the end of J** (or 1-2 weeks later)
M starts **when everything else has finished**. N after or at same time as M [9]

(ii) week number: 26

Allow f.t. [1]

2 **(a)**
```
parent(ali, ahmed).
parent(meena, ahmed).
```

Accept statements in either order
Wrong capitalisation minus 1 mark [2]

(b)
```
P =
ahmed
aisha
```

Ignore capitalisation
Deduct 1 mark for every extra result [2]

(c)
```
mother(M, gina).
```

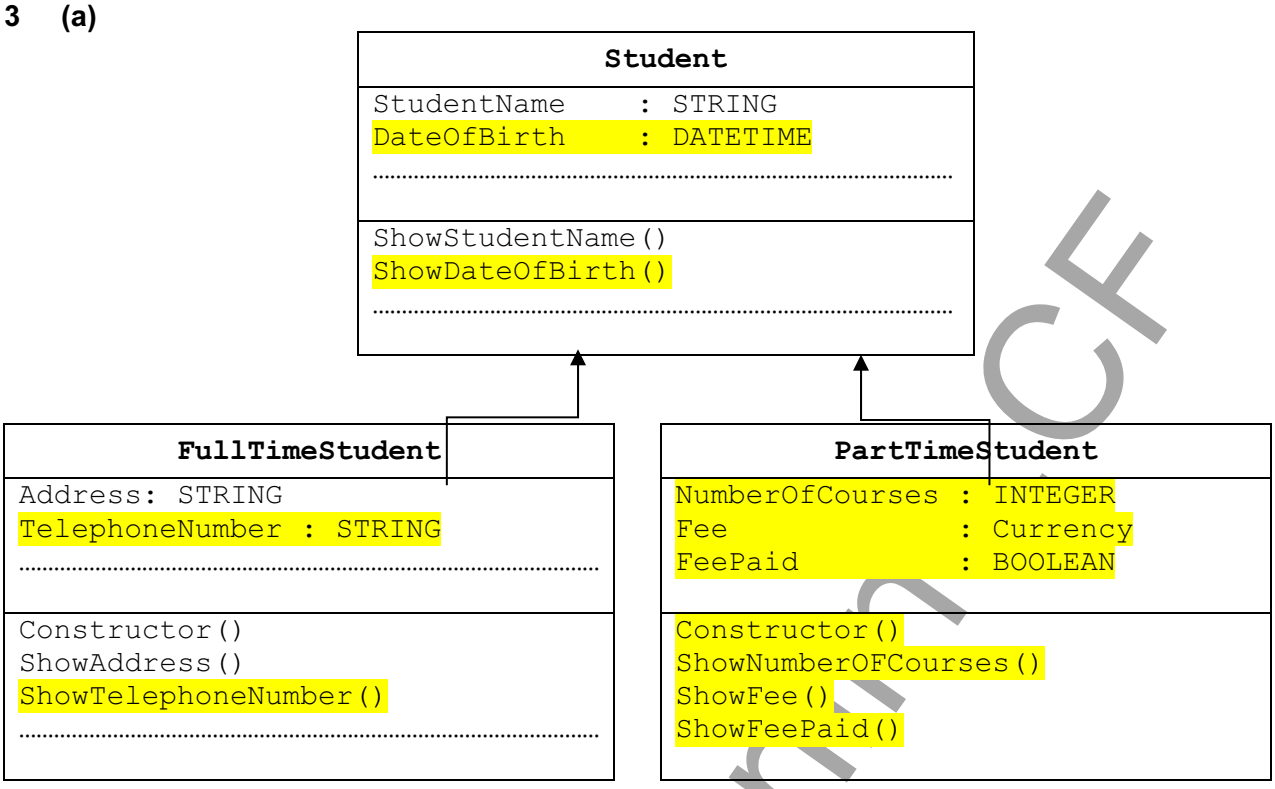
Accept `parent(M, gina) AND female(M).` Accept a comma instead of `AND`
Reject `mother(M, gina) IF female(M) AND parent(M, gina).`
Ignore capitalisation [1]

(d)
```
father(F, C)
IF
male(F) AND parent(F, C).
```
(1) (1) [2]

(e)
```
brother(X, Y)
IF
male(X)AND                    [1]
parent(A, X)  AND             [1]
parent(A, Y)                  [1]
AND NOT X=Y.                  [1]
```

Accept any variable for A, but it must be the same in both places
Accept father/mother instead of parent
Ignore capitalisation

3 (a)



Mark as follows:

Base class:
- dateOfBirth declaration and associated method in Student
- constructor

Subclasses:
- telephoneNumber declaration and associated method in FullTimeStudent
- NumberOFCourses declaration and associated method in PartTimeStudent
- fee declaration and associated method in PartTimeStudent
- feepaid declaration and associated method in PartTimeStudent
- constructor method in PartTimeStudent
- inheritance arrows

Ignore data types, ignore other methods/attributes
Ignore brackets after methods [Max 7]

(b) (i) Mark as follows (parts to be ignored in grey):

If no programming language stated, map to 1 of the 3 below (or check in Q1ai)
Class header & ending (watch out these may be squashed into the next clip)
Ignore methods
2 attributes with correct data types
No mark if subclass properties shown here
Attributes required:
StudentName
DateOfBirth (accept variations e.g. DoB)

Pascal

```
TYPE Student = CLASS
   PUBLIC
      Procedure ShowStudentName();
      Procedure ShowDateOfBirth();
   PRIVATE
      StudentName : STRING;
      DateOfBirth : TDateTime;  // accept string    reject Date
END;
```

Python

```
class Student :
    def __int__(self) :
        self.__StudentName = ""
        self.__DateOfBirth = ""  # date(1,1,2015)
    def ShowStudentName() :
        pass
    def ShowDateOfBirth() :
        pass
```

Ignore __ before attributes

VB.NET

```
Class Student
    Public Sub ShowStudentName()
    End Sub
    Public Sub ShowDateOfBirth()
    End Sub
    Private StudentName As String
    Private DateOfBirth As Date     ' accept string
End Class
```

(Ignore: must inherit)
Ignore Private/protected/public
Don't give a mark if using DIM [2]

(ii) Mark as follows:

– Class header and showing superclass
– Properties (Do not award this mark if properties from base class included here)
  Data types must be correct
– Methods (Do not award this mark if methods from base class included here)
  must show heading and ending of procedure/function declaration
  Ignore PUBLIC, PRIVATE

Pascal

```
TYPE FullTimeStudent = CLASS (Student)
   PUBLIC
      Procedure ShowAddress();
      Procedure ShowTelephoneNumber();
   PRIVATE
      Address         : STRING;
      TelephoneNumber : STRING; // reject integer
END;
```

Python

```
class FullTimeStudent(Student) :
    def __init__(self) :
        self.__Address = ""
        self.__TelephoneNumber = ""
    def ShowAddress() :
        pass
    def ShowTelephoneNumber() :
        pass
```

VB.NET

```
Class FullTimeStudent : Inherits Student
    Public Sub ShowAddress()
    End Sub
    Public Sub ShowTelephoneNumber()
    End Sub
    Private Address As String
    Private TelephoneNumber As String  ' reject integer
End Class
```

No mark if using `DIM` [3]

(iii) 1 mark per statement to max 3
Missing string delimiters: penalise once
Accept use of constructor

Pascal

```
NewStudent := FullTimeStudent.Create;
NewStudent.StudentName := 'A.Nyone';
NewStudent.DateOfBirth := EncodeDate(1990, 11,12);//:=
'11/12/1990'
NewStudent.TelephoneNumber := '099111';
```

Alternative

```
NewStudent := FullTimeStudent.Create('A.Nyone', '12/11/1990',
'099111');
```

Python

```
NewStudent = FullTimeStudent()
NewStudent.StudentName = "A.Nyone"
NewStudent.DateOfBirth = "12/11/1990"
NewStudent.TelephoneNumber = "099111"
```

Alternative

```
NewStudent = FullTimeStudent('A.Nyone', '12/11/1990', '099111')
```

VB.NET

```
Dim NewStudent As FullTimeStudent = New FullTimeStudent()
NewStudent.StudentName = "A.Nyone"
NewStudent.DateOfBirth = #11/12/1990#
NewStudent.TelephoneNumber = "099111"
```

Alternative

```
Dim NewStudent As FullTimeStudent = New
FullTimeStudent("A.Nyone", "12/11/1990", "099111")
```

[Max 3]

4 (a)

```
FUNCTION Hash(Key : STRING) RETURNS INTEGER
DECLARE Number : INTEGER
Number ← ASCII(LEFTSTRING(Key,1))
// Number ← ASCII(Key[1])
Number ← Number - 64
RETURN Number
// Result ← Number  // Hash ← Number
ENDFUNCTION
```

Accept `ASC` instead of `ASCII`
Accept `LEFT` instead of `LEFTSTRING`
Key can be a different identifier but must be the same in both places [5]

(b) (i)

Dictionary

Index	Key	Value
1		
2		
3	Computer	Rechner
4	Disk	Platte
5	Error	Fehler
6	File	Datei
7		
8		
:	:	
:	:	
1999		
2000		

Ignore spelling mistakes
1 mark for 2 correct pairs entered in correct slots [2]

(ii) Collision / synonym / space already occupied / same index in array
Overwrites previous key-value pair

reject error [Max 2]

(iii) Create an overflow area
The 'home' record has a pointer to others with the same key // linked list
OR
Store the overflow record at the next available address …
in sequence (= next available)
OR
Re-design the hash function …. // write a different/another algorithm
to generate a wider range of indexes // enlarging storage space // to create fewer collisions [2]

(iv) Mark as follows:

Check whether slot is empty:
`IF Dictionary[Index,1] <>"" // != '' // > NULL // > NONE`
If not: update index: `THEN Index ← <some value>`
…to find an empty slot (loop / follow pointer / go to overflow area) reject FOR loop
Insert code between lines 20 and 30

```
21 WHILE Dictionary[Index,1] > ""
22   Index ← Index + 1
23   IF Index > 2000
24     THEN
25       Index ← 1
26   ENDIF
27 ENDWHILE
```
[4]

5 (a) (i)

Accumulator	Memory Address 509	510	511	512
0	7	3	0	0
7				7
0				
1			1	
7				
14				14
1				
2			2	
14				
21				21
2				
3			3	
3 marks			*1 mark*	*1 mark*

If values changed in column 509 or 510 don't give marks for 511/512 [5]

(ii) stores the counter value for …./// acts as a control variable/counter
How many times the loop has been performed // control the loop

Ignore re-stating the steps [2]

(b) LDM #12 (must be instruction before storage)
STO 509 (must be final instruction)

1 mark for each instruction [2]

6 (a) 1 mark for structure header/ending
1 mark for each field correct, take away 1 mark for additional fields
Python answers will use a class

Pascal

```
TYPE StockItem = RECORD
   ProductCode     : String;   // accept integer
   Price           : Currency; // accept real
   NumberInStock   : Integer;
END;
```

Python

```
class StockItem :
    def __init__(self) :
        self.ProductCode = ""     # = 0
        self.Price = 0.0          # = 0
        self.NumberInStock = 0
```

VB.NET

```
STRUCTURE StockItem
    Dim ProductCode As String     ' accept integer
    Dim Price As Decimal          ' Double/single
    Dim NumberInStock As Integer
END STRUCTURE
```

VB6

```
Type StockItem
    ProductCode As String     ' accept integer
    Price As Currency         ' Double/single
    NumberInStock As Integer
END Type
```

[4]

(b) (i)
```
01 TRY
02    OPENFILE "StockFile" FOR READ/RANDOM   // ignore "
03 EXCEPT
04    OUTPUT "File does not exist"
05 ENDTRY
```
[2]

(ii) (Line 01) alerts system to check for possible run-time errors (exception)
(Lines 03, 04) handle the exception without the program crashing // keeps program running// provide alternative statements to execute to avoid run-time error

Accept "exception handling" for 1 mark [Max 2]

(c)
```
WHILE NOT EOF("StockFile")
   READFILE "StockFile", ThisStockItem // accept reading separate
fields
   OUTPUT ThisStockItem.ProductCode
   OUTPUT ThisStockItem.NumberInStock
ENDWHILE
```

1 mark for loop (accept REPEAT)
1 mark for EOF("StockFile") // StockFile.Peek <> -1 / NONE/"
1 mark for READ record
1 mark for OUTPUT of 2 fields

Ignore opening and closing file [4]