

CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the May/June 2015 series

9608 COMPUTER SCIENCE

9608/23

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 May/June 2015 series for most Cambridge IGCSE[®], Cambridge International A and AS Level components and some Cambridge O Level components.

® IGCSE is the registered trademark of Cambridge International Examinations.

Page 2	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9608	23

1 (a)

Identifier	Data Type	Description
HorseName	STRING	Name of the horse
NumberOfPreviousWins	INTEGER	Number of previous wins
RacePenaltyWeight	INTEGER / REAL / SINGLE	Penalty weight

[1]

(b) (i) Stepwise refinement // top-down design

[1]

(ii) INPUT HorseName
 INPUT NumberOfPreviousWins
 RacePenaltyWeight ← 0
 IF NumberOfPreviousWins = 1 OR NumberOfPreviousWins = 2
 THEN
 RacePenaltyWeight ← 4
 ENDIF
 IF NumberOfPreviousWins > 2
 THEN
 RacePenaltyWeight ← 8
 ENDIF
 OUTPUT HorseName, RacePenaltyWeight

Mark as follows:

(OUTPUT) + INPUT x 2 (1 mark)
 Two/three conditions in evidence correctly formed (1 mark)
 (penalise Assignment used for equals)
 Condition for penalty weight = 0 + assignment = 0 (1 mark)
 Other conditions X 2 + Assignment of 4 and 8 (1 mark)
 Final output of horse name + penalty weight (1 mark)

[5]

2 (a) (i) 7

[1]

(ii) 2
 9

[2]

(b) (i)

Input value	Output			Comment
	Fifty Dollar	Twenty Dollar	Ten Dollar	
70	1	1	0	Least possible number of notes
85	(0	0	0)	Error message
130	2	1	1	Least possible number of notes
600	(0	0	0)	Error message

Penalise any number entries on the 85 and 600 rows

[3]

(ii) INPUT **Amount**

IF Amount > 500

THEN

OUTPUT "Refused - amount too large"

ELSE

IF (Amount MOD 10) <> 0 / >0

THEN

OUTPUT "Refused - not a multiple of \$10"

ELSE

FiftyDollar ← Amount DIV 50

Temp ← Amount MOD 50 //

(Amount - 50 * FiftyDollar)

TwentyDollar ← Temp DIV 20 //

(Amount MOD 50) DIV 20

Temp ← Temp MOD 20

TenDollar ← Temp DIV 10

ENDIF

ENDIF

[max 5]

3 (i)

A	Width	in any order
B	Length	
C	JobID	
D	CustomerName	in any order
E	JobCost	

[5]

Page 4	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9608	23

- (ii) PROCEDURE CalculateJobCost
 (BYREF JobCost : INTEGER/CURRENCY/REAL,
 BYVALUE Length : INTEGER,
 BYVALUE Width : INTEGER)

mark as follows:

identifier + data type × 3	(3 marks)	
jobcost (only) BYREF	(1 mark)	
length, width (only) BYVALUE/BYREF	(1 mark)	[5]

- 4 (a) (i) ERROR [1]
 (ii) parityerrorcheck [1]
 (iii) Binary Coded Decimal // Binary ▼ Coded ▼ Decimal [2]

- (b) (i) OPENFILE "DISPENSERS" FOR WRITE (1 mark)
REPEAT (1 mark)
 OUTPUT "Enter dispenser code (XXXXX to end)"
 INPUT DispenserCode
 IF DispenserCode <> "XXXXX"
 THEN
 OUTPUT "Enter bank code ..."
 INPUT BankCode
 LineString ← CONCAT(DispenserCode, "▼", BankCode) (1 mark)
 // now write the new line to the file
WRITEFILE ("DISPENSERS"), LineString (1 mark)
 ENDIF
 UNTIL DispenserCode = "XXXXX" (1 mark)
CLOSE ("DISPENSERS") // **CLOSEFILE** (1 mark)
 OUTPUT "DISPENSERS file now created" [6]

- (ii) • Bank code/ Dispenser code is digit characters only
 • Bank code is exactly 3 digits // Dispenser code is exactly 5 digits
 • Range check on Bank code between 1 and 999
 // range check on dispenser code between 1 and 99999

Note: If no reference made to either Bank code or Dispenser code MAX 1 [max 2]

- (iii) data of the existing 15 dispensers will be lost/overwritten [1]
 (iv) Append // Illustrated with program code statement [1]

Page 5	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9608	23

(c) *Mark as follows:*

- Variables declared/commented (at least X2) (1 mark)
- Input of 'ThisBank' with prompt (1 mark)

- File open statement (1 mark)
- File mode is 'Input' (1 mark)
- File close

- Loop (Not a FOR loop) (1 mark)
- Until all records considered

- Isolate LineBankCode (1 mark)
- Isolate LineDispenserCode

- Count initialised (1 mark)
- Count incremented (1 mark)

- Output – List of dispenser codes (1 mark)
- Output – dispenser count (1 mark)

[max 10]

Visual Basic ...

```

Dim DispenserRecord As String
Dim DispenserCode As String : Dim Bank As String
Dim DispenserCount As Integer
Dim ThisBank As String
FileOpen(1, "C:\DISPENSERS.txt", OpenMode.Input)

Console.WriteLine()
Console.Write("Which bank ..(Three digit code)? ")
ThisBank = Console.ReadLine

DispenserCount = 0
Do
    DispenserRecord = LineInput(1)
    DispenserCode = Left(DispenserRecord, 5)
    Bank = Mid(DispenserRecord, 7, 3)

    If Bank = ThisBank Then
        DispenserCount = DispenserCount + 1
        Console.WriteLine(DispenserCode)
    End If
Loop Until EOF(1)
FileClose(1)

Console.WriteLine()
Console.WriteLine("There are " & DispenserCount & " dispensers
for this bank")

```

Page 6	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9608	23

Python ...

```
# DispenserLine      - String
# DispenserCode     - String
# Bank              - String
# DispenserCount    - Integer
# ThisBank          - String

MyFile = open("c:\DISPENSERS.txt", "r")

ThisBank = input("Which bank ..(Three digit code)? ")

DispenserCount = 0
while 1:
    DispenserLine = MyFile.readline()
    if not DispenserLine:
        break
    DispenserCode = DispenserLine[0:5]
    # slices chars 0,1,2,3,4
    Bank = DispenserLine[6:9] # slices chars 6,7,8

    if Bank == ThisBank:
        DispenserCount = DispenserCount + 1
        print(DispenserCode)

MyFile.close()
print
print("There are " + str(DispenserCount)
" dispensers for this bank")
```

Pascal ...

```
var DispenserRecord : String ;
var DispenserCode  : String ;
var Bank           : String ;
var DispenserCount : Integer ;
var ThisBank       : String ;
var TheFile        : Text ;

begin
assign(TheFile, 'K:\DISPENSERS.txt') ;
reset(TheFile) ;

WriteLn() ;
Write('Which bank ..(Three digit code)? ') ;
ReadLn(ThisBank) ;
C
DispenserCount := 0 ;
repeat
    readln(TheFile, DispenserRecord) ;
    DispenserCode := Copy(DispenserRecord,1, 5) ;
    Bank := copy(DispenserRecord, 7, 3) ;

    If Bank = ThisBank Then
        begin
            DispenserCount := DispenserCount + 1 ;
```

Page 7	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9608	23

```

        WriteLn(DispenserCode)
    end ;

    until EOF(TheFile) ;
    close(TheFile) ;

writeLn() ;
writeLn('Dispenser count: ', DispenserCount) ;

readLn ;
end.

```

- 5 (a) (i) • Set of data items have a common name (1 mark)
- Items are referenced using a subscript/index (1 mark)
- Accept: all data items are of the same data type (1 mark) [max 2]
- (ii) 24 [1]
- (iii) • The total number of amplifiers 'produced' by workers 1, 2 and 3/three workers (1 mark)
- on day 2_ (1 mark) [2]

(b)

WorkerNum	DayNum	WorkerAverage	OUTPUT	WorkerTotal		
				1	2	3
1				0		
2					0	
3						0
1	1			10		
	2			21		
	3			31		
	4			45		
2	1				20	
	2				36	
	3				60	
	4				80	
3	1					9
	2					20
	3					33
	4					50
1		2.25				
2		2				
3		1.25	INVESTIGATE 3			

[8]

Page 9	Mark Scheme	Syllabus	Paper
	Cambridge International AS/A Level – May/June 2015	9608	23

- (c) (i) WorkerNum : INTEGER (1 mark)
DayNum : INTEGER (1 mark)
WorkerTotal : ARRAY OF INTEGER
(1 mark) (1 mark)
WorkerAverage : REAL (1 mark) [max 4]

(ii) PROCEDURE AnalyseProductionData (NumDays : INTEGER, NumWorkers : INTEGER)

```

FOR WorkerNum ← 1 TO 3
  WorkerTotal [WorkerNum] ← 0
ENDFOR

FOR WorkerNum ← 1 TO 3
  FOR DayNum ← 1 TO 4
    WorkerTotal [WorkerNum] ← WorkerTotal [WorkerNum] +
      ProductionData [WorkerNum, DayNum]
  ENDFOR
ENDFOR

FOR WorkerNum ← 1 TO 3
  WorkerAverage = WorkerTotal [WorkerNum] / (4) *
  DailyHoursWorked [WorkerNum]
  IF WorkerAverage < 2
    THEN
      OUTPUT "Investigate" WorkerNum
    ENDIF
  ENDFOR

ENDPROCEDURE

```

Mark as follows:

All '3's changed to NumWorkers

All '4's changed to NumDays

WorkerAverage '4' changed to NumDays [3]

(iii) (CALL) AnalyseProductionData (7, 13) [1]