CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the October/November 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 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 **(i)** 40 [1]

(ii) 314.2(0) [1]

(iii) 16 // ERROR as identifier Z has not been declared [1]

(iv) TRUE [1]

2 **(i)** (Single) software program [1]
Features for:
program editor/writing/editing
translation // interpreter/compiler
testing program code **//** observe outputs } 2 points to score [1]

(ii) Syntax checking (on entry)
Structure blocks (e.g. IF structure and loops begin/end highlighted)
General prettyprint features
Automatic indentation
Highlights any undeclared variables
Highlights any unassigned variables
Commenting out/in of blocks of code
Visual collapsing / highlighting of blocks of code
Single stepping
Breakpoints
Variable/expressions report window [MAX 3]

3 (a)

	Inputs		Output	
Test Case	InA	InB	OutZ	
1	TRUE	TRUE	**FALSE**	[1]
2	TRUE	FALSE	**TRUE**	[1]
3	FALSE	TRUE	**TRUE**	[1]
4	FALSE	FALSE	**TRUE**	[1]

(b)
```
IF InA = TRUE AND InB = TRUE
   THEN
      OutZ ← FALSE
   ELSE
      OutZ ← TRUE
ENDIF
```

Mark as follows

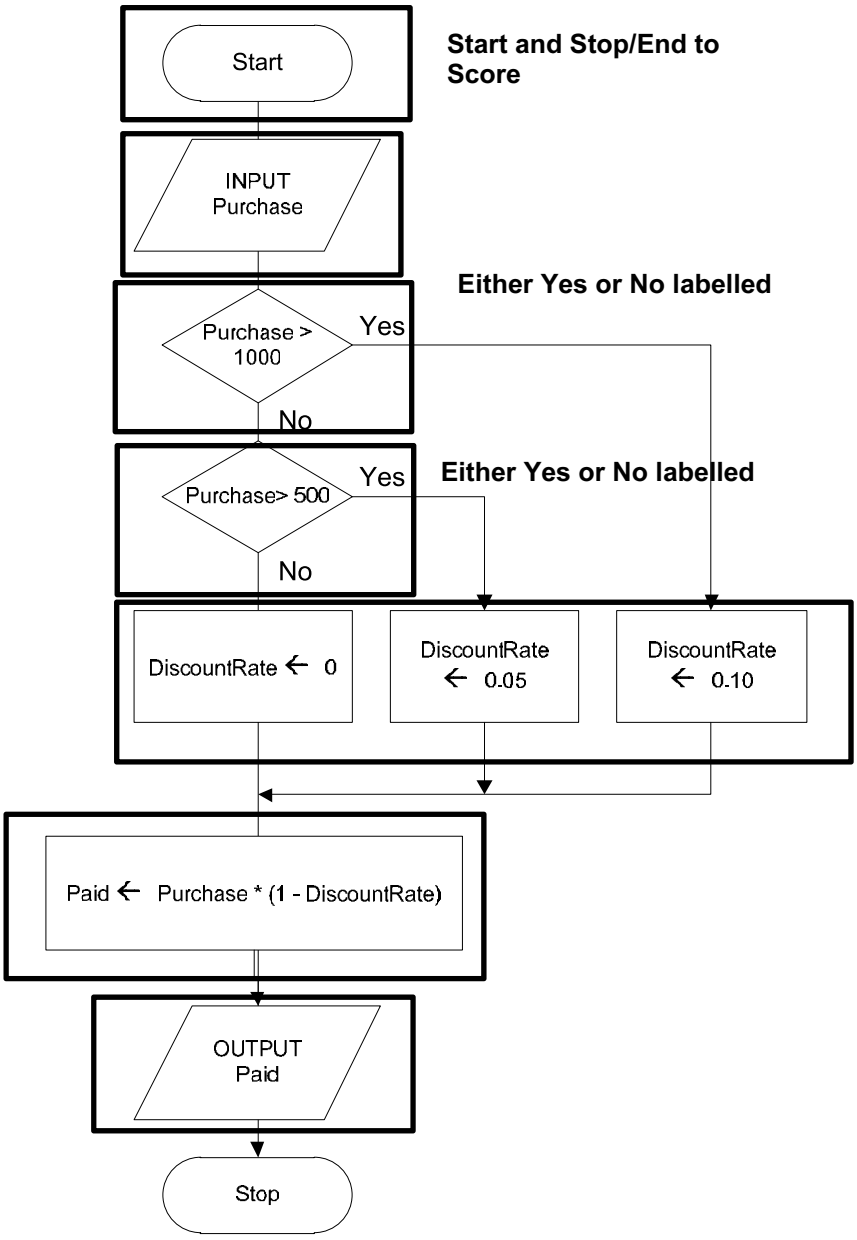
Structure: `IF – THEN – ELSE – ENDIF` [1]

Condition: `InA = TRUE AND InB = TRUE` [1]

Logic:
`OutZ ← FALSE (when condition true)`
`OutZ ← TRUE (when condition false)` } [1]

Alternative answer (worth 3 marks):
```
OutZ ← NOT(InA AND InB)
OutZ ← NOT InA OR NOT InB
```

4



Start and Stop/End to Score

Either Yes or No labelled

Either Yes or No labelled

[MAX 6]

[Total: 6]

[MAX 6]

5 (a)

Identifier	Data type	Description	
YearCount	INTEGER	Loop counter /// Age of the car	[1]
PurchasePrice	INTEGER	Purchase price of the car	[1]
CurrentValue	REAL // CURRENCY **Allow: SINGLE** **Refuse: DOUBLE**	The changing depreciated value	[1]

Must have correct identifier + Data type + Description to score

(b)

```
OUTPUT "Enter Purchase price"
INPUT PurchasePrice
CurrentValue ← PurchasePrice                                    [1]
YearCount ← 1
WHILE YearCount < 9 AND CurrentValue >= 1000                    [2]
      Note: Penalise: inclusion of $
   IF YearCount = 1                                             [1]
      THEN
         CurrentValue ← CurrentValue * (1 - 40/100)
      ELSE
         CurrentValue ← CurrentValue * 0.8                      [1]
   ENDIF
   OUTPUT YearCount,  CurrentValue
   YearCount  ← YearCount + 1                                   [1]
ENDWHILE
```

6 (a) Combination of staff and task number // the pair of numbers // the pair of random numbers [1]
//there will be duplicates /repeats//some staff tasks will not be generated [1]

(b) (i) `04 // 03` [1]

(ii) `27 // 28` [1]

(iii) `20` [1]

(iv) `11 / 12` [1]

(c) (i) Zero [1]

(ii) `Completed <> 60 // NewStaffTask = FALSE` [1]
Allow: Inclusion of the WHILE

(iii) Determines whether this combination of `StaffNum` and `TaskNum` has been completed [1]
Assigns value TRUE if not already generated [1]
Flags that this is the first time this staff + task has been selected/to exit the loop [1]
Outputs the new staff + task number [1]

[MAX 3]

(iv) `TaskGrid : ARRAY[1:5, 1:12] OF BOOLEAN`
1 mark | 1 mark [2]

(d) ***Pseudocode …***

`(SELECT) CASE (OF) + ENDCASE` using `StaffNo`
1 mark 1 mark

```
(CASE) 1: StaffName ← "Sadiq"                    1
(CASE) 2: StaffName ← "Smith"
(CASE) 3: StaffName ← "Ho"
(CASE) 4: StaffName ← "Azmah"
(CASE) 5: StaffName ← "Papadopolis"
                                  (all four cases …)   1
ENDCASE // ENDSELECT
```
[4]

Visual Basic

```
Select Case StaffNo
   Case 1
      StaffName = "Sadiq"
   Case 2
      StaffName = "Smith"
   Case 3
      StaffName = "Ho"
   Case 4
      StaffName = "Azmah"
   Case 5
      StaffName = "Papadopolis"
End Select
```

7 (a) (i) CAT [1]
Ignore any opening + closing quotes

(ii) 13 [1]

(iii) 83 [1]

(iv) 15 [1]

(b) *Input of string …*
Correct syntax (for both prompt and assignment)
Uses `MyString` identifier } [1]

`StringTotal` set to 0 [1]

FOR loop:
`FOR - NEXT` keywords // (Python) correct indentation [1]
Correct start and /end boundaries [1]
Note: the end boundary must use the language length function/method // alternative Python syntax

Isolate single character number [1]

Followed by the use of Asc (VB) // Ord (Python)
Assigned to `NextNum` } [1]

Added to `StringTotal` [1]

Correct syntax for the output of the string and number [1]

[MAX 6]

Python …

```
MyString = input('key in string')

StringTotal = 0
for i in range (0, len(MyString)):
   NextNum = ord(MyString[i])
   StringTotal = StringTotal + NextNum

print(MyString, StringTotal)
```

Visual Basic…

```
Dim MyString As String
Dim StringTotal As Integer
Dim i As Integer
Dim NextNum As Integer

Console.Write("key in string")
MyString = Console.ReadLine
StringTotal = 0
For i = 1 To Len(MyString) // MyString.Length
   NextNum = Asc(Mid(MyString, i, 1))
   StringTotal = StringTotal + NextNum
Next

Console.WriteLine(MyString & "  " & Str(StringTotal))
```

Pascal …

```
VAR MyString     : String ;
VAR StringTotal  : Integer ;
VAR i            : Integer ;
VAR NextNum      : Integer ;
VAR SingleChar   : Char;

begin
   Writeln('key in string');
   readln(MyString) ;
   StringTotal := 0 ;

   For i := 1 To Length(MyString) do
      begin
         SingleChar := MyString[i] ;
         NextNum := Ord(SingleChar) ;
         StringTotal := StringTotal + NextNum  ;
      end ;

   WriteLn(MyString, StringTotal) ;

   ReadLn() ;
End.
```

(c) Used to provide an integrity/verification check [1]
Used as a checksum [1]
The total can be recalculated by the receiving software [1]
If any of the characters have been incorrectly transmitted the recalculated total and transmitted total will not match [1]
[MAX 2]

8 (a) r [1]
Ignore inclusion of any quotes

(b) (i) 2 [1]
Ignore inclusion of any quotes for part (i), (ii) and (iii)

(ii) + [1]

(iii) 7 [1]

(c) (i)

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT
2	5	3	8	40	-	1	1/40

[2]

(ii)

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT
3	4	1	4	16	+	16	1

[2]

(iii)

N1	N2	N3	N4	BottomAnswer	Op	TopAnswer	OUTPUT
7	9	2	3	27	+	39	
						12	
							1 12/27

[3]

(d) (i) Adaptive (maintenance) [1]

(ii) Allow more than two fractions to be added [1]
Numerator/denominator more than 1 digit [1]
Multiply and division also possible [1]
Allow brackets [1]
Give answer as decimal number [1]
Lowest possible denominator [1]
Trap any fraction which has a zero numerator [1]
Allow the input of vulgar fraction(s) [1]

[MAX 3]