CAMBRIDGE INTERNATIONAL EXAMINATIONS

Cambridge International Advanced Subsidiary and Advanced Level

MARK SCHEME for the May/June 2015 series

9608 COMPUTER SCIENCE

9608/21 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.

CAMBRIDGE
International Examinations

1 (a)

Identifier	Data Type	Description
`RaceHours`	`INTEGER`	The hours part of the race time
`RaceMinutes`	`INTEGER`	the minute part of the race time
`RaceSeconds`	`INTEGER // REAL`	the seconds part of the race time
`RaceTime`	`INTEGER // REAL`	the race time in seconds

3 × (meaningful name + data type) [3]

(b) (i)

Identifier	Data Type	Description
`PersonalBestTime`	`INTEGER/REAL`	Personal best time in seconds

meaningful name + data type [1]

(ii) Mark as follows:

- Declarations/comments for variables – at least 2
- Input (+ prompts) for hours, minutes, seconds
- Input (+ prompt) of personal best time
- Correct calculation of `RaceTimeInSeconds` (don't allow use of 'x' for '*')
- Output `RaceTimeInSeconds`
- Correct logic and output message for < personal best
- Correct logic and output message for > personal best
- Correct logic and output message for = personal best [max 7]

(c) (i)

- Choosing data/values…
- Test every possible 'logic path' through the code
  // with knowledge of the structure/code

Ignore any reference to normal/boundary/extreme … [2]

(ii)

- `PersonalBest` column labelled
- Test number 1 message: "Equals personal best time"/or similar
- Test 2/Test 3 – data for better performance …
- Described with suitable message
- Test 2/Test 3 – data for worse performance …
- Described with suitable message [6]

2 (a) (i) Displays the menu (choices)
Repeats the prompt and input …
…the input is a number between 1 and 4 // Checks number is between 1 and 4

"within range" is not enough [3]

(ii) …the input number is validated [1]

(b) (i) 3 [1]

(ii) Previous design repeated indefinitely // (new design) limits number of attempts

Penalise "Program terminates/closes" [1]

(c)
```
IF Choice = 1 THEN (CALL) ReadFile                    (1)
IF Choice = 2 THEN OUTPUT "Add Customer code"         (1)
IF Choice = 3 THEN OUTPUT "Search Customer code"      (1)
IF Choice = 4 THEN END                                (1)
```

alternative answer:

mark as follows:

```
CASE OF Choice // Select CASE Choice      1 mark
1: (CALL) ReadFile                        1 mark (allow CASE = 1)
2: OUTPUT "Add Customer code"             1 mark
3: OUTPUT "Search Customer code"          1 mark
4: END
ENDCASE
```

Output strings must match [max 3]

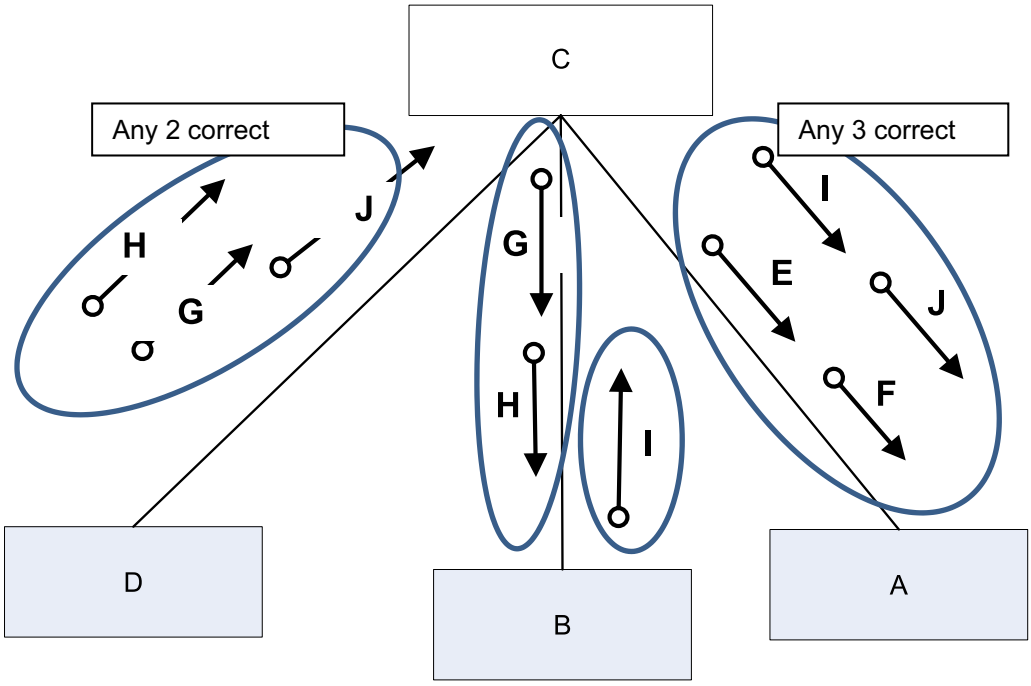
(d) Mark as follows:

- `Choice / NoOfAttempts` declared/commented as integer
  Must appear within the 'main' program
  Allow: different identifier names
- Constant `i` assigned a value 3
- There is an 'outer' loop to repeatedly display the menu
- Input 'choice' variable
- Three `IF` statements (or equivalent) for processing menu choices 1, 2 and 3
  Note: they must be correctly formed as 'nested' or 'independent'
- Choice 1 calls procedure `ReadFile`
- Choice 2 outputs "Add Customer Code"
  + Choice 3 outputs "Search Customer Code"
- Outer loop terminates correctly with '`Choice` = 4' //or equivalent
- Procedure `DisplayMenu` shows the four menu options
- Procedure `ReadFile` is present …
  and contains a single output message 'Read file code' [max 8]

3 (a) Control box – C // Produce insurance quotation [1]

D // Input customer details + A // Send quotation letter is correct positions [1]

(b)



Data items	
E	CustomerName
F	CustomerEmail
G	Model
H	Age
I	PolicyCharge
J	PolicyNumber

[4]

4 (i)

```
FOR NoOfThrows ← 1 TO 20 / 0 TO 19
      1              1                                    (2)
   INPUT Player1Throw
   INPUT Player2Throw                                     (1)
   IF Player1Throw > Player2Throw
      THEN
      Player1Total ← Player1Total + 1                     (1)
   ENDIF
   IF Player2Throw > Player1Throw
      THEN
         Player2Total ← Player2Total + 1
   ENDIF
ENDFOR                                                    (1)

IF Player1Total > Player2Total
   THEN
      OUTPUT "Player1 is the winner"
   ELSE
      OUTPUT "Player2 is the winner"
END
```

[5]

(ii) Player scores equal // if `Player1Total` = `Player2Total` // there is no winner // a draw [1]

5 (a)

- 1D Array // List [1]
- INTEGER [1]

(b) (i)

x	DayNumber	OUTPUT
0	1	
	2	
1	3	5/6/2015
	4	
2	5	7/6/2015
	6	
3	7	9/6/2015
		3

Note: 'x' and 'output' entries must be on or below the relevant 'DayNumber' entry
Mark as above [4]

(ii)
- … Sales for the first seven days (1)
- … the number of days on which the total sales were **10 or over** (1)
- Outputs the corresponding dates (1)
- Output the final value/total (of x) (1) [max 3]

(c) (i) 2 [1]

(ii)

Tick Cross	Explanation (if invalid)
X // ✓	2nd parameter should be `CHAR` // accept just tick
X	Three parameters/should be 2 parameters
✓	

[3]

(d)
```
OPENFILE "DISCOUNT_DATES" FOR WRITE / WRITING      (1)
INPUT NextDate                                     (1)
WHILE  NextDate <> "XXX"
   INPUT Discount
   NextLine = CONCAT(NextDate, " ", Discount)      (1)
   WRITEFILE "DISCOUNT_DATES",  NextLine
ENDWHILE                                           (1)
OUTPUT "File now created"
CLOSEFILE
```
[4]

(e) (i) Sensible Identifier **+** Data Type **+** Description (1 + 1 + 1)

For example:

`ThisDate`	`STRING/DATE`	date 'entered by user'
`Found`	`BOOLEAN`	flag to indicate `ThisDate` is 'present in the file'
`NextLine`	`STRING`	a single line 'from the text file'
`NextDate`	`STRING/DATE`	date 'from next line in the file'
`NextDiscount`	`STRING`	the discount value from `NextLine`
`ThisMonth`	`INTEGER`	the month part of the date (input or from file)
`MyStreamReader`	`STREAMREADER`	references `DISCOUNT_DATES` file

Reject 'generic' reserved words
Allow **one** instance variable to store output string(s)
Allow **one** instance of month/day/year number e.g. `ThisMonth` shown above [3]

(ii) *Mark as follows:*

Open file statement	(1)
File read statement for line text – `NextLine`	(1)
File close statement	(1)
Input of the required date – `ThisDate`	(1)
Isolate `NextDate` from `NextLine`	(1)
Isolate `NextDiscount` from `NextLine`	(1)
IF statement comparing the two dates	(1)
Uses Boolean variable `Found` to flag when found	(1)
Post/pre condition loop iterate through the file	(1)
Test for <u>EOF or 'found'</u>	(1)
Note: These must follow some correct logic to score …	
Output 'No discount on this date' **and** Output 'This is a discount date')	(1)
Output (when date not found) 'Date not found'	(1)

Accept 'any' identifier names [max 7]

APPENDIX
Programming Solutions

Question 1 (b) (ii)

Visual Basic …

```
Dim RaceHours As Integer
Dim RaceMinutes As Integer
Dim RaceSeconds As Integer
Dim RaceTimeInSeconds As Integer
Dim PersonalBest As Integer

Console.Write("Time in hours ... ") : RaceHours = Console.ReadLine
Console.Write("Time in minutes... ") : RaceMinutes = Console.ReadLine
Console.Write("Time in seconds ... ")
RaceSeconds = Console.ReadLine
Console.Write("Personal best in seconds ... ")
PersonalBest = Console.ReadLine
RaceTimeInSeconds = RaceHours*60*60 + RaceMinutes*60 + RaceSeconds
Console.Write(RaceTimeInSeconds)
If RaceTimeInSeconds < PersonalBest Then
  Console.WriteLine("New personal best time")
Else
  If RaceTimeInSeconds = PersonalBest Then
   Console.WriteLine("Equals personal best time")
  Else
   Console.WriteLine("Below personal best")
  End If
End If
```

Python …

```
# RaceHours         - Integer
# RaceMinutes       - Integer
# RaceSeconds       - Integer
# RaceTimeInSeconds - Integer
# PersonalBest      - Integer

RaceHours = int(input("Time in hours ... "))
RaceMinutes = int(input("Time in minutes... "))
RaceSeconds = int(input("Time in seconds ... "))
PersonalBest = int(input("Personal best in seconds ... "))

RaceTimeInSeconds = RaceHours*60*60 + RaceMinutes*60 + RaceSeconds

if RaceTimeInSeconds < PersonalBest:
   print("New personal best time")
elif RaceTimeInSeconds == PersonalBest:
   print("Equals personal best time")
else:
   print("Below personal best")
```

Programming Solutions
Question 1 (b) (ii) – contd.

Pascal …

```
var RaceHours         : Integer ;
var RaceMinutes       : Integer ;
var RaceSeconds       : Integer ;
var RaceTimeInSeconds : Integer ;
var PersonalBestTime  : Integer ;

begin
Writeln('Time in hours ... ') ; readln(RaceHours) ;
Writeln('Time in minutes... ') ; readln(RaceMinutes) ;
Writeln('Time in seconds ... ')  ;
readln(RaceSeconds) ;
Writeln('Personal best in seconds ... ') ;

Readln(PersonalBest) ;
RaceTimeInSeconds := RaceHours*60*60 + RaceMinutes*60 + RaceSeconds  ;
Writeln(RaceTimeInSeconds) ;

If RaceTimeInSeconds < PersonalBestTime Then
  WriteLn('New personal best time')
  Else
   If RaceTimeInSeconds = PersonalBest Then
      WriteLn('Equals personal best time')
   Else
      WriteLn('Personal best time is unchanged) ;

Readln;
End
```

Programming Solutions
Question 2 (d)

Visual Basic …

```
    Dim Choice As Integer
    Dim NoOfAttempts As Integer
    CONST i = 3
    Do
       Call DisplayMenu()
       NoOfAttempts = 0
       Do
       Console.Write("Enter choice (1..4)"
       Choice = Console.ReadLine
       NoOfAttempts = NoOfAttempts + 1
       Loop Until (Choice >= 1 And Choice <= 4) Or NoOfAttempts = i

       If Choice = 1 Then Call ReadFile()
       If Choice = 2 Then Console.WriteLine("Add customer code")
       If Choice = 3 Then Console.WriteLine("Search customer code")
    Loop Until Choice = 4

  Sub DisplayMenu()
   Console.WriteLine()
   Console.WriteLine("1. Read customer file")
   Console.WriteLine("2. Add customer")
   Console.WriteLine("3. Search for a customer")
   Console.WriteLine("4. End")
   Console.WriteLine()
  End Sub

  Sub ReadFile()
   Console.WriteLine("Read file code")
  End Sub
```

Python …

```
def DisplayMenu():
    print()
    print("1. Read customer file")
    print("2. Add customer")
    print("3. Search for a customer")
    print("4. End")
    print()

def ReadFile():
  print("Read file code”)

if __name__ ==”__main__” :
    # Choice - Integer
    # NoOfAttempts - Integer

    Choice = 0
    while Choice !=4:
       DisplayMenu()
```

```
Choice = int(input”Enter choice (1..4) :”)
NoOfAttempts = 1
while (Choice < 1 or Choice >4) and NoOfAttempts < 3:
   Choice = int(input”Enter choice (1..4) :”)
   NoOfAttempts = 1
if Choice == 1:
   ReadFile()
elif Choice == 2:
   print("Add customer code")
elif Choice == 3:
print("Print customer code")
```

Programming Solutions
Question 2 (d) – contd.

Pascal …

```
var Choice       : Integer ;
var NoOfAttempts : Integer ;
const i = 3 ;

procedure DisplayMenu ;
  begin
  WriteLn();
  WriteLn('1. Read customer file') ;
  WriteLn('2. Add customer')  ;
  WriteLn('3. Search for a customer');
  WriteLn('4. End')  ;
  WriteLn() ;
End ;

Procedure ReadFile ;
  begin
  WriteLn('Read file code');
End ;

begin

repeat
  DisplayMenu() ;
  NoOfAttempts := 0 ;
  repeat
   Writeln('Enter choice (1..4)') ; ReadLn(Choice) ;
   NoOfAttempts := NoOfAttempts + 1 ;
  Until ((Choice >= 1) And (Choice <= 4)) Or (NoOfAttempts = i);

  If Choice = 1 Then ReadFile() ;
  If Choice = 2 Then writeLn('Add customer code');
  If Choice = 3 Then WriteLn('Search customer code') ;
Until Choice = 4 ;

end.
```

Programming Solutions
Question 5 (ii)

Visual Basic …

```
Dim ThisDate As String : Dim NextDate As String
Dim FileString As String
Dim Found As Boolean

FileOpen(1, "D:DISCOUNT_DATES.txt", OpenMode.Input)
       or equivalent for a 'StreamReader' solutions

Console.Write("Date to find (DD/MM/YYYY)..")
ThisDate = Console.ReadLine
Found = False
Do
   FileString = LineInput(1)
   NextDate = Left(FileString, 10)
   If NextDate = ThisDate Then
Found = True
' length is 15 when shows TRUE
If Len(FileString) = 15 Then
       Console.WriteLine("This is a discount date")
   Else
       Console.WriteLine("No discount on this date")
 End If
 End If
 Loop Until Found = True Or EOF(1)

 FileClose(1)

 If Found = False Then
   Console.WriteLine("Date not found")
 End If
```

Python …

```
MyFile = open("c:\DISCOUNT_DATES.txt", "r")
ThisDate = input("Next date ...(XXX to end)")

Found = 0
while Found == 0:
   NextLine = MyFile.readline()
   if not NextLine:
      break

   FileDate = NextLine[0:10]
   DiscountIndicator = NextLine[11:]

   if FileDate == ThisDate:
        Found = 1
       print (ThisDate, DiscountIndicator)

MyFile.close()
if Found == 0:
```

```
   print ("This date was not found")
```

Note: `Found` could be Boolean to give:

```
Found = False
while not Found:
```

Programming Solutions
Question 5 (ii) – contd.

Pascal …

```
var ThisDate   : String ;
var NextDate   : String ;
var TheFile    : Text ;
var FileString : String ;
var Found      : Boolean ;

begin
assign(TheFile, 'k:\DISCOUNT_DATES.txt') ;
reset(TheFile) ;

writeln('Date to find (DD/MM/YYYY)..')  ;
readln(ThisDate) ;
Found := False ;

repeat
  readln(TheFile, FileString) ;
  NextDate := copy(FileString,1, 10) ;

  If NextDate = ThisDate then
   begin
   Found := True  ;
   { length is 15 when shows TRUE }
   if length(FileString) = 15 then
      writeLn('This is a discount date')
   else
      writeLn('No discount on this date')
      end ;

until Found = True or EOF(TheFile) ;

close(TheFile) ;

if Found = False then writeLn('Date not found') ;
```