Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/23**

Paper 2 Written Paper **May/June 2016**

MARK SCHEME

Maximum Mark: 75

Published

This document consists of **12** printed pages.

CAMBRIDGE International Examinations

1 (a) (i) **[6]**

Item	Statement	Selection	Iteration	Assignment
1	`WHILE DegF > 37.5`		✓	
2	`MyName = "Gordon"`			✓
3	`DegF = INT(DegF)`			✓
4	`ENDIF`	✓		
5	`CASE OF MyFavourite`	✓		
6	`UNTIL x = 5`		✓	

One mark per row
Additional ticks in any row cancels that row

(ii) **[6]**

Item	Purpose of statement
1	(Start of) loop – repeat while `DegF` greater than `37.5`
2	Assign (string) "`Gordon`" to `MyName`
3	Assign integer value / whole number part of `DegF` to `DegF`
4	End of an `IF` statement / selection statement
5	Head of `CASE` / selection statement based on variable `MyFavourite`
6	End of `REPEAT` / post-condition loop: repeated until `x` equals `5`

Exact wording not important
Explanation must refer to variables or values used in code (except for row 4)

(iii) **[2]**

Expression	Result
`'P' & MID(MyString, 13, 4)`	**"Paint"**
`RIGHT(MID(MyString, 6, 10), 4)`	**"Main"**

Must have correct case
Quotation marks optional

2 (a) **[5]**

Identifier	Data Type	Description
`Accelerator`	`INTEGER`	Accelerator position Values: 0 to 100 in steps of 1 Meaning: 0 : none (not pressed) 100: maximum (fully pressed)
`EngineTemp`	`REAL / FLOAT / SINGLE / DOUBLE`	Engine temperature in °C (–50 to +150 correct to 1 decimal place)
`NormalTemp`	`INTEGER`	Normal engine temperature in °C Whole number; typical value 90
`Speed`	`INTEGER`	Road speed of car (in km/hr) Values: 0 to 200 in steps of 1
`EngineStop`	`BOOLEAN`	Value used to signal engine must be stopped Possible values: TRUE: stop engine FALSE: run engine

One mark per row
Data types as shown

(b) **[6]**

```
   INPUT Accelerator
   INPUT EngineTemp      ①
   INPUT NormalTemp
   INPUT Speed
                                                   ③
   IF Accelerator = 0 AND EngineTemp >= NormalTemp AND Speed = 0
      THEN
②        EngineStop ← TRUE      ④

      ELSE
         EngineStop ← FALSE     ⑤
   ENDIF

   OUTPUT "Engine Stopped"      ⑥
```

Mark points as circled, descriptions as below:

1. Four INPUT statements (correct names and sequence)
2. Correct `IF..THEN..ELSE..ENDIF` including first condition (or equivalent nested `IF`s)
3. Correct second and third conditions
4. Correct `THEN` statement
5. Correct `ELSE` statement
6. Output indicating EngineStop
   (Following `ENDIF` or as **two** separate statements within `THEN` and `ELSE`)

3 (a) **[11]**

```
FUNCTION Decrypt (Lookup : ARRAY, CipherChar : CHAR) RETURNS CHAR
   DECLARE Found          : BOOLEAN
   DECLARE Index          : INTEGER
   DECLARE OriginalChar   : CHAR

   Index ← 1              //start with first element in the array //
                           assign the start index
   Found ← FALSE
      //now search for CipherChar in Looookup:
   WHILE Found = FALSE // Found <> TRUE // NOT Found
       // compare CipherChar with this array element:
     IF Lookup[Index] = CipherChar
       THEN
         Found ← TRUE         // Set the flag
       ELSE
         Index ← Index + 1    //Move to next array element
     ENDIF
   ENDWHILE
   //dropped out of loop so must have found CipherChar:
   OriginalChar ← CHR(Index)  // convert Index to original character
   RETURN OriginalChar

ENDFUNCTION
```

One mark for each part-statement (shown underlined and bold)

(b) 'Pseudocode' solution included here for development and clarification of the mark scheme. Programming language example solutions appear in the Appendix. [6]

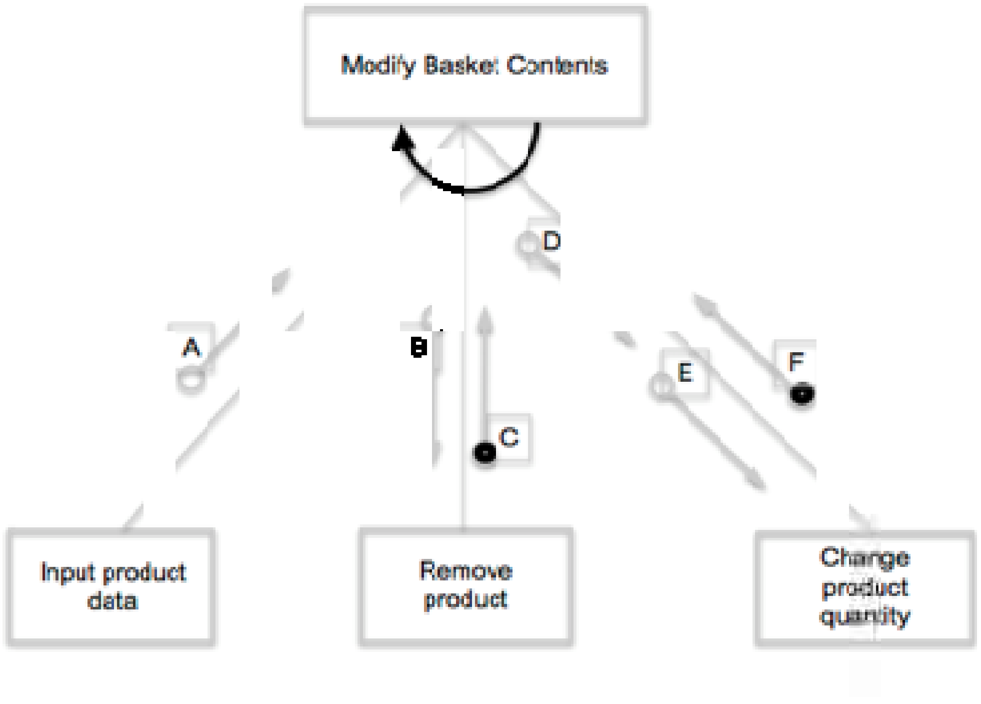
```
INPUT StartIndex                 } 1
INPUT NumberToOutput

FOR Index ← StartIndex to StartIndex + NumberToOutput - 1

   OriginalChar ← CHR(Index) 3

   CipherChar ← Lookup[Index] 4
                                                               5
  OUPUT ("Index " & Index & ": Character " & OriginalChar &
         " has substitute character " & CipherChar) 6

ENDFOR
```

Mark points as circled, descriptions as below:

1. Two INPUT statements
2. Working loop using Index (allow alternative solutions including separate loop counter)
3. Assignment (using correct values of Index or other variable)
4. Assignment (using correct values of Index or other variable)
5. One mark for OUTPUT of a string combining text and variables...
6. ...a second mark if OUTPUT string is completely correct

4 (a)
- Functions / Procedures **[2]**
- Ability to pass parameters between modules
- Use of local / global variables

(b) (i) **[1]**



One mark for correct arrow as shown – accept either direction

(ii) **[4]**

Data Item	**Parameter**					
	A	**B**	**C**	**D**	**E**	**F**
Product ID	✓	✓		✓	(✓)	
Quantity				(✓)	✓	
Flag Value – indicating operation success or fail			✓			✓

Mark as follows:
Row 1: One mark for tick in A **AND** B, one mark for D **OR** E
Row 2: One mark for D **OR** E (must be opposite of Row 1)
Row 3: One mark for C **AND** F

5 (a) (i) **Explanation**: **[Max 3]**

- Easier to separate the two strings // to retrieve / search for / edit (text relating to a CD)
- Obvious where `CDTitle` ends and `CDArtist` begins

Drawback:

- Takes up more / unnecessary space in the file
- If the string is bigger than 40 characters then data will be lost // string length is limited
- The additional spaces will need to be removed before strings can be used

One mark per bullet

(ii) **Problem**: File mode = `WRITE` / file is opened for writing // by explanation **[3]**

Effect: All existing file lines / contents / data will be overwritten / deleted / lost

Solution: `WRITE` should be changed to `APPEND` (allow meaningful example)

Allow first two mark points to be interchanged – read as one paragraph.

(b) 'Pseudocode' solution included here for development and clarification of mark scheme. Programming language example solutions appear in the Appendix. **[Max 10]**

```
PROCEDURE OutputLocationList()

  DECLARE CDTitle : STRING
  DECLARE CDArtist : STRING
  DECLARE CDSearch : STRING
  DECLARE FileData : STRING
  DECLARE Total : INTEGER
  DECLARE FileData : STRING

  Total ← 0

  OPENFILE "MyMusic" FOR READ

  OUTPUT "Input Location"
  INPUT CDSearch

  WHILE NOT EOF("MyMusic")
    READFILE ("MyMusic", FileData
    IF RIGHT(FileData, 8) = CDSearch
      THEN
        CDTitle ← LEFT(FileData, 40)
        CDArtist ← MID(FileData(41, 40)
        OUPUT (CDTitle & " - " & CDArtist)
     Total ← Total + 1
    ENDIF
  ENDWHILE

  OUTPUT (Total & " CDs found"
  CLOSEFILE("MyMusic")

ENDPROCEDURE
```

One mark for each of the following:

- Procedure heading and ending
- Declaration AND initialisation of variable used as counter (`Total` above)
- Prompt and input of location to search for `CDSearch` (or other name)
- Open file for reading (Allow `MyMusic` or `MyMusic.txt`)
- Working conditional loop structure including test for `EOF`

The following points must be present inside a loop
- Read a line from the file (or read complete file in one e.g. as list)
- Isolate 8 chars representing location **and** compare with `CDLocation`
- Extract strings representing `CDTitle` and `CDArtist`
- Output `CDTitle` and `CDArtist` (separator optional) if correct location found
- Increment `Total` if correct location found

The following points must be present after a loop
- Output message including number of CDs found at location (`Total`)
- Close file

6 (a) (i) **[4]**

n	x	Flag	m	NewString
–		TRUE	7	“”
1	‘B	FALSE		“B”
2	‘i’			“Bi”
3	‘g’			“Big”
4	‘▿’	→ TRUE		“Big▿“
5	‘B’	← FALSE		“Big▿B”
6	‘e’			“Big▿Be”
7	‘n’			“Big▿Ben”

One mark per correct column (**columns 3 and 4 count as 1**)

- Arrows indicate required sequence (i.e. “B” can’t be in row before TRUE)
- Letter case must be as shown
- Ignore quotation symbol
- Allow “ “ or ▿ for space symbol

(ii) (To return a string where:) **[2]**

- The first character of each word is capitalised / made upper case // by explanation
- The remaining characters (of each word) are made lower case

(b) (i) • The function operates normally // returns an ‘empty string’ **[1]**

(ii) Examples of suitable test strings: **[3]**

- Strings with all capitals
- Strings with all lower case
- Strings with first letters of words already capitalised (i.e. in correct format)
- Strings in “reverse” format – i.e. first letters lower case, the rest upper case
- String with only one word
- Strings with multiple spaces
- Strings with numbers / symbols

One mark for each string example **plus** supporting explanation.

Appendix – Program Code Example Solutions

Q3 (b): VB.NET

```
Console.WriteLine("Enter start position")
StartIndex = Console.ReadLine()
Console.WriteLine("Enter how many")
NumberToOutput = Console.ReadLine()
For Index = StartIndex To StartIndex + NumberToOutput - 1
   OriginalChar = Chr(Index)
   CipherChar = Lookup(Index)
   Console.WriteLine("Index " & Index & ": Character " & OriginalChar &
                    " has substitute character " & CipherChar)
Next Index
```

Q3 (b): Pascal

```
Writeln('Enter start position');
Readln(StartIndex);
Writeln('Enter how many');
Readln(NumberToOutput);
For index := StartIndex To StartIndex + NumberToOutput – 1 Do
Begin
   OriginalChar := chr(index);
   CipherChar := Lookup[index];
   writeln("Index " + index + ": Character " + OriginalChar +
           " has substitute character " & CipherChar);
end;
```

Q3 (b): Python

```
startIndex = int(input("enter start position"))
numberToOutput = int(input("enter how many"))
for index in range(startIndex, (startIndex + numberToOutput)) :
   OriginalChar = chr(index)
   CipherChar = Lookup[index]
   print("Index " + (index) + ": Character " + OriginalChar +
         " has subst char " + CipherChar)
```

Q5 (b): VB.NET

A StreamReader() solution:

```
Sub OutPutLocationList()

   Dim Total As Integer
   Dim FileData As String
   Dim ObjReader As IO.StreamReader
   ObjReader = New IO.StreamReader("C:\MyMusic.txt")
   Dim CDLocation As String
   Dim CDTitle As String
   Dim CDArtist As String

   Total = 0
   Console.WriteLine("Input location to search ")
   CDSearch = Console.ReadLine
   Do While ObjReader.Peek <> -1
      FileData = ObjReader.ReadLine()
      If Right(FileData, 8) = CDLocation Then
         CDTitle = Left(FileData, 40)
         CDArtist = Mid(FileData, 41, 40)
         Console.WriteLine(CDTitle & "   " & CDArtist)
         Total = Total + 1
      End If
   Loop
   Console.WriteLine(Total & " CDs were found")
   ObjReader.Close()
End Sub
```

A legacy FileOpen() solution:

```
Sub OutPutLocationList()

   Dim Total As Integer
   Dim FileData As String
   FileOpen (1, "C:\MyMusic.txt", OpenMode.Input)
   Dim CDLocation As String
   Dim CDTitle As String
   Dim CDArtist As String

   Total = 0
   Console.WriteLine("Input location to search ")
   CDSearch = Console.ReadLine
   Do While NOT EOF(1)
      Input(1, FileData)
      If Right(FileData, 8) = CDSearch Then
         CDTitle = Left(FileData, 40)
         CDArtist = Mid(FileData, 41, 40)
         Console.WriteLine(CDTitle & "   " & CDArtist)
         Total = Total + 1
      End If
   Loop
   Console.WriteLine(Total & " CDs were found")
  FileClose(1)
End Sub
```

Q5 (b): Pascal

```
procedure OutputLocationList;
var
   FileData, CDLocation, CDTitle, CDArtist : String;
   CDFile : Textfile;
   Total : Integer;
Begin
   Total := 0;
   Writeln('Input location to search ');
   Readln(CDSearch);
   AssignFile(CDFIle, 'MyMusic.txt');
   Reset(CDFile);

   While not eof(CDFile) Do
   Begin
     readln(CDFile, FileData);
     If copy(FileData, 80, 8) = CDSearch Then
     Begin
        CDTitle := copy(FileData, 1, 40);
        CDArtist := copy(FileData, 41, 40);
        Writeln(CDTitle + ‘ : ‘ + CDArtist);
        Total := Total + 1;
     End;
   End;

   Writeln(Total + ‘ CDs were found’)
   CloseFile(CDFile);

End
```

Q5 (b): Python

```
#total  : Integer
#CDSearch, LineOfText, LineString  : String

Def OutPutLocationList():
   FileHandle = open("MyMusicPy.TXT", "r")
   total = 0
   CDSearch = input("Enter location to search")
   LineOfText = FileHandle.readline()
   while len(LineOfText) > 0:
      LineString = LineOfText[80:87]    #extact last 8 characters
(location)
      if LineString == CDSearch:
         total = total+1
         CDTitle = LineOfText[0:39]    #extract CD title
         CDArtist = LineOfText[40:79]  #extract CD artist
         print(CDTitle + ": " + CDArtist)
         LineOfText = FileHandle.readline()  #read next line
   print("There are " + str(total) + " in that location")
   FileHandle.close()
```