Cambridge International AS & A Level

Cambridge International Examinations
Cambridge International Advanced Subsidiary and Advanced Level

COMPUTER SCIENCE **9608/43**

Paper 4 Written Paper **May/June 2017**

MARK SCHEME

Maximum Mark: 75

Published

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 2017 series for most Cambridge IGCSE®, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>
<table>
<tr><th>Label</th><th>Op code</th><th>Operand</th><th>Comment</th><th></th></tr>
<tr><td>START:</td><td>IN</td><td></td><td>// INPUT character</td><td rowspan="2">1</td></tr>
<tr><td></td><td>STO</td><td>CHAR</td><td>// store in CHAR</td></tr>
<tr><td></td><td>LDM</td><td>#65</td><td>// Initialise ACC (ASCII value for 'A' is 65)</td><td>1</td></tr>
<tr><td>LOOP:</td><td>OUT</td><td></td><td>// OUTPUT ACC</td><td>1 + 1</td></tr>
<tr><td></td><td>CMP</td><td>CHAR</td><td>// compare ACC with CHAR</td><td>1</td></tr>
<tr><td></td><td>JPE</td><td>ENDFOR</td><td>// if equal jump to end of FOR loop</td><td>1</td></tr>
<tr><td></td><td>INC</td><td>ACC</td><td>// increment ACC</td><td>1</td></tr>
<tr><td></td><td>JMP</td><td>LOOP</td><td>// jump to LOOP</td><td>1</td></tr>
<tr><td>ENDFOR:</td><td>END</td><td></td><td></td><td></td></tr>
<tr><td>CHAR:</td><td colspan="2"></td><td></td><td></td></tr>
</table>
</td><td>8</td></tr>
<tr><td>1(b)</td><td>
<table>
<tr><td>START:</td><td>LDD</td><td>NUMBER</td><td></td><td>1</td></tr>
<tr><td></td><td>AND</td><td>MASK</td><td>// set to zero all bits except sign bit</td><td>1</td></tr>
<tr><td></td><td>CMP</td><td>#0</td><td>// compare with 0</td><td>1</td></tr>
<tr><td></td><td>JPN</td><td>ELSE</td><td>// if not equal jump to ELSE</td><td>1</td></tr>
<tr><td>THEN:</td><td>LDM</td><td>#80</td><td>// load ACC with 'P' (ASCII value 80)</td><td>1</td></tr>
<tr><td></td><td>JMP</td><td>ENDIF</td><td></td><td></td></tr>
<tr><td>ELSE:</td><td>LDM</td><td>#78</td><td>// load ACC with 'N' (ASCII value 78)</td><td rowspan="2">1</td></tr>
<tr><td>ENDIF:</td><td>OUT</td><td></td><td>//output character</td></tr>
<tr><td></td><td>END</td><td></td><td></td><td></td></tr>
<tr><td>NUMBER:</td><td colspan="2">B00000101</td><td>// integer to be tested</td><td></td></tr>
<tr><td>MASK:</td><td colspan="2">B10000000</td><td>// show value of mask in binary here</td><td>1</td></tr>
</table>
</td><td>7</td></tr>
</table>

Question	Answer	Marks
2(a)	1 mark for the declaration of the array. 1 mark for assigning a 0 to Customer ID (CustomerID ← 0) 1 mark for getting the correct record (Customer[x].) 1 mark for setting up a loop to go <u>from 0 to 199</u> `DECLARE Customer : ARRAY[0 : 199] OF CustomerRecord` 1 `FOR x ← 0 TO 199` 1 `Customer[x].CustomerID ← 0` 1+1 `ENDFOR`	**4**

2(b)(i)	`PROCEDURE InsertRecord(BYVAL NewCustomer : CustomerRecord)`
	`  TableFull ← FALSE`
	`  // generate hash value`
	`  Index ←` **`Hash(NewCustomer.CustomerID)`** 1
	`  Pointer ← Index  // take a copy of index`
	
	`  // find a free table element`
	`  WHILE` **`Customer[Pointer].CustomerID > 0`** 1
	`  Pointer ←` **`Pointer + 1`** 1
	`  // wrap back to beginning of table if necessary`
	`  IF` **`Pointer > 199`** 1
	`  THEN`
	`Pointer ← 0` 1
	`  ENDIF`
	`  // check if back to original index`
	`  IF` **`Pointer = Index`** 1
	`  THEN`
	`  TableFull ← TRUE`
	`  ENDIF`
	`  ENDWHILE`
	`  IF` **`NOT TableFull`** 1
	`  THEN`
	`Customer[Pointer] ← NewCustomer` 1
	`  ELSE`
	`OUTPUT "Error"` 1
	`  ENDIF`
	`ENDPROCEDURE`
		9

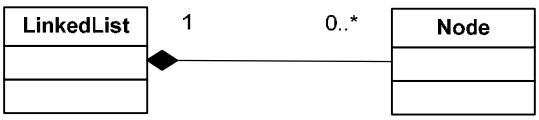
Question	Answer	Marks

2(b)(ii)	FUNCTION SearchHashTable(BYVAL SearchID : INTEGER) RETURNS INTEGER
	  // generate hash value
	  Index ← **Hash(SearchID)** 1
	  // check each record from index until found or not there
	  WHILE (**Customer[Index].CustomerID <> SearchID)** 1
	  AND (**Customer[Index].CustomerID > 0**) 1
	  Index ← Index + 1 1
	  // wrap if necessary
	  IF **Index > 199** 1
	  THEN
	  Index ← 0 1
	  ENDIF
	  ENDWHILE
	  // has customer ID been found?
	
	  IF **Customer[Index].CustomerID = SearchID** 1
	  THEN
	  RETURN Index 1
	  ELSE
	  RETURN -1 1
	  ENDIF
	ENDFUNCTION
		9

2(b)(iii)	A record out of place may not be found	**1**

Question	Answer	Marks

3	`FUNCTION Find(BYVAL Name : STRING,`
	`  BYVAL Start : INTEGER,`
	`  BYVAL Finish : INTEGER) RETURNS INTEGER`
	`  // base case`
	`  IF` **`Finish < Start`** 1
	`  THEN`
	`  RETURN -1`
	`  ELSE`
	`  Middle ←` **`(Start + Finish) DIV 2`** 1
	`  IF` **`NameList[Middle] = Name`** 1
	`  THEN`
	`  RETURN` **`Middle`** 1
	`  ELSE  // general case`
	`  IF SearchItem >` **`NameList[Middle]`** 1
	`  THEN`
	`Find(Name, Middle + 1, Finish)` 1
	`  ELSE`
	`Find(Name, Start, Middle - 1)` 1
	`  ENDIF`
	`  ENDIF`
	`  ENDIF`
	`ENDFUNCTION`
		7

Question	Answer	Marks
4(a)(i)	containment/aggregation	**1**
4(a)(ii)	 1 mark for the two classes (in boxes) and connection with correct end point 1 mark for 0 ..* 0	**Max 2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(b)</td><td>

mark as follows:

- Class heading and ending
- Constructor heading and ending
- Parameters in constructor heading
- Declaration of (private) attributes : Pointer, Data
- Assignment of parameters to Pointer and Data

Python Example

<pre>
class Node: 1
   def __init__(self, D, P):   1 + 1
      self.__Data = D   1
      self.__Pointer = P   1
      return
</pre>

Example Pascal

<pre>
type
   Node = class   1
      private   1
         Data : String;
         Pointer : Integer;
      public
         constructor Create(D : string; P : integer);
         procedure SetPointer(P : Integer);
         procedure SetData(D : String);
         function GetData() : String;   ignore
         function GetPointer() : Integer;
   end;
  constructor Node.Create(D : string; P : integer);   1+1
   begin
      Data := D;
      Pointer := P;   1
   end;
</pre>

Example VB.NET

<pre>
Class Node   1
   Private Data As String
   Private Pointer As Integer   1
   Public Sub New(ByVal D As String, ByVal P As Integer)   1+1
      Data = D
      Pointer = P   1
   End Sub
End Class
</pre>

</td><td>5</td></tr>
<tr><td>4(c)(i)</td><td>A pointer that doesn't point to any data/node/address</td><td>1</td></tr>
</table>

Question	Answer	Marks
4(c)(ii)	–1 (accept NULL) The array only goes from 0 to 7 // the value is not an array index	**2**

4(c)(iii)	**mark as follows:**
	• Class and constructor heading and ending
	• Declare private attributes (HeadPointer, FreeListPointer, NodeArray)
	• Initialise HeadPointer to null
	• Initialise FreeListPointer to 0
	• Looping 8 times …
	• Creating empty node in NodeArray
	• Use .SetPointer method to point each new node to next node
	• Set last node pointer to null pointer
	
	Python Example
	`class LinkedList:` 1
	`  def __init__(self):` 1
	`  self.__HeadPointer = - 1` 1
	`  self.__FreeListPointer = 0` 1
	`  self.__NodeArray = []`
	`  for i in range(8):` 1
	`  ThisNode = Node("", (i + 1))` 1
	`  self.__NodeArray.append(ThisNode)`
	`  self.__NodeArray[7].SetPointer(- 1)` 1
	
	Example Pascal
	`type`
	`LinkedList = class` 1
	`  private`
	`  HeadPointer : Integer;`
	`  FreeList : Integer;`
	`  NodeArray : Array[0..7] of Node;`
	`  public`
	`  constructor Create();`
	`  procedure FindInsertionPoint(NewData : string; var PreviousPointer, NextPointer : integer);`
	`  procedure AddToList(NewData : string);`
	`  procedure OutputListToConsole();`
	`end;`
	`constructor LinkedList.Create();` 1
	`var i : integer;`
	`begin`
	`  HeadPointer := -1;` 1
	`  FreeList := 0;` 1
	`  for i := 0 To 7 do` 1
	`  NodeArray[i] := Node.Create('', (i + 1));` 1
	`  NodeArray[7].SetPointer(-1);` 1
	`end;`
		Max 7

Question	Answer	Marks

	Example VB.NET
	`Class LinkedList` 1
	`  Private HeadPointer As Integer`
	`  Private FreeList As Integer`
	`  Private NodeArray(7) As Node`
	
	`  Public Sub New()` 1
	`  HeadPointer = -1` 1
	`  FreeList = 0` 1
	`  For i = 0 To 7` 1
	`  NodeArray(i) = New Node("", (i + 1))` 1
	`  Next`
	`  NodeArray(7).SetPointer(-1)` 1
	`  End Sub`
	`End Class`	

4(c)(iv)	• Creating instance of LinkedList assigned to contacts
	
	Python Example
	`contacts = LinkedList()`
	
	Pascal Example
	`var contacts : LinkedList;`
	`  contacts := LinkedList.Create;`
	
	VB.NET Example
	`Dim contacts As New LinkedList`
		1

Question	Answer	Marks

4(c)(v)	**mark as follows:**
	• Start with HeadPointer
	• Output node data
	• Loop until null pointer
	• Following pointer to next node
	• Use of getter (ie GetData/GetPointer)
	
	Python Example
	`def OutputListToConsole(self) :`
	`  Pointer = self.__HeadPointer` 1
	`  while Pointer != -1 :` 1
	`  print(self.__NodeArray[Pointer].GetData())` 1+1
	`  Pointer = self.__NodeArray[Pointer].GetPointer()` 1
	`  print()`
	`  return`
	
	Pascal Example
	`procedure LinkedList.OutputListToConsole();`
	`  var Pointer : integer;`
	`  begin`
	`  Pointer := HeadPointer;` 1
	`  while Pointer <> -1 do` 1
	`  begin`
	`  WriteLn(NodeArray[Pointer].GetData);` 1+1
	`  Pointer := NodeArray[Pointer].GetPointer;` 1
	`  end;`
	`  end;`
	
	VB.NET Example
	
	`Public Sub OutputListToConsole()`
	`  Dim Pointer As Integer`
	`  Pointer = HeadPointer` 1
	`  Do While Pointer <> -1` 1
	`  Console.WriteLine(NodeArray(Pointer).GetData)` 1+1
	`  Pointer = NodeArray(Pointer).GetPointer` 1
	`  Loop`
	`End Sub`
		5

Question	Answer	Marks

4(c)(vi)	**mark as follows:**
	• Store free list pointer as NewNodePointer
	• Store new data item in free node
	• Adjust free pointer
	• F list is currently empty
	• Make the node the first node
	• Set pointer of this node to Null Pointer
	• Find insertion point
	• If previous pointer is Null pointer
	• Link this node to front of list
	• Link new node between Previous node and next node
	
	Python Example
	`def AddToList(self, NewData):`
	
	`  NewNodePointer = self.__FreeListPointer`
	
	`  self.__NodeArray[NewNodePointer].SetData(NewData)`
	
	`  self.__FreeListPointer = self.__NodeArray[self.__FreeListPointer].GetPointer()`
	
	`  if self.__HeadPointer == -1:`
	
	`  self.__HeadPointer = NewNodePointer`
	`  self.__NodeArray[NewNodePointer ].SetPointer(-1)`
	`  else:`
	`  PreviousPointer, NextPointer = self.FindInsertionPoint(NewData)`
	`  if PreviousPointer == -1 :`
	
	`  self.__NodeArray[NewNodePointer ].SetPointer (self.__HeadPointer)`
	`  self.__HeadPointer = NewNodePointer`
	`  else:`
	
	`  self.__NodeArray[NewNodePointer ].SetPointer(NextPointer)`
	`  self.__NodeArray[PreviousPointer].SetPointer(NewNodePointer)`
		Max 6

Question	Answer	Marks
	Pascal Example	

```
    procedure LinkedList.AddToList(NewData : string);
    var NewNodePointer , PreviousPointer,
                                        NextPointer : integer;
    begin
      // make a copy of free list pointer
      NewNodePointer := FreeListPointer;
      // store new data item in free node
      NodeArray[NewNodePointer].SetData(NewData);
      // adjust free pointer
      FreeListPointer :=
NodeArray[FreeListPointer].GetPointer;
      // if list is currently empty
      if HeadPointer = -1
         then
            // make the node the first node
            begin
               HeadPointer := NewNodePointer;
               // set pointer to Null pointer
               NodeArray[NewNodePointer].SetPointer(-1);
            end
         else
            // find insertion point
            begin
               FindInsertionPoint(NewData, PreviousPointer,
                                                 NextPointer);
               // if previous pointer is Null pointer
               if PreviousPointer = -1
                  then
                     // link node to front of list
                     begin
                        NodeArray[NewNodePointer]
                                    .SetPointer(HeadPointer);
                        HeadPointer := NewNodePointer ;
                     end
                  else
                     // link new node between
                                  Previous node and next node
                     begin
                        NodeArray[NewNodePointer ]
                                      .SetPointer(NextPointer);
                        NodeArray[PreviousPointer]
                                 .SetPointer(NewNodePointer);
                     end;
             end;
     end;
```

Question	Answer	Marks

	VB.NET Example
	`Public Sub AddToList(ByVal NewData As String)`
	`  Dim NewNodePointer, PreviousPointer, NextPointer As Integer`
	`  ' make copy of free list pointer`
	`  NewNodePointer= FreeListPointer`
	`  ' store new data item in free node`
	`  NodeArray(NewNodePointer).SetData(NewData)`
	`  ' adjust free pointer`
	`  FreeListPointer = NodeArray(FreeListPointer).GetPointer`
	`  ' if list iscurrently empty`
	`  If HeadPointer = -1 Then`
	`  ' make the node the first node`
	`  HeadPointer = NewNodePointer`
	`  ' set pointer to Null pointer`
	`  NodeArray(NewNodePointer).SetPointer(-1)`
	`  Else`
	`  ' find insertion point`
	`  FindInsertionPoint(NewData, PreviousPointer, NextPointer)`
	`  ' if previous pointer is Null pointer`
	`  If PreviousPointer = -1 Then`
	`  ' link to front of list`
	`  NodeArray(NewNodePointer).SetPointer(HeadPointer)`
	`  HeadPointer = NewNodePointer`
	`  Else`
	`  ' link new node between Previous node and next node`
	`  NodeArray(NewNodePointer).SetPointer(NextPointer)`
	`  NodeArray(PreviousPointer).SetPointer(NewNodePointer)`
	`  End If`
	`  End If`
	`End Sub`	

Question	Answer	Marks

Pseudocode for reference:

```
PROCEDURE AddToList(NewData)
   // remember value of free list pointer
   NewNodePointer ← FreeListPointer
   // add new data item to free node pointed to by free list
   NodeArray[NewNodePointer].Data ← NewData
   // adjust free pointer to point to next free node
   FreeListPointer  ← NodeArray[FreeList].Pointer
   // is list currently empty?
   IF HeadPointer = NullPointer
      THEN
         // make the node the first node
         HeadPointer ← NewnodePointer
         // set pointer of new node to Null pointer
         NodeArray[NewNodePointer].Pointer ← NullPointer
      ELSE
         // find insertion point
         CALL FindInsertionPoint(NewData, PreviousPPointer, NextPointer)
         // if previous pointer is Null pointer
         IF PreviousPointer = NullPointer
            THEN
               // link new node to front of list
               NodeArray[NewNodePointer].Pointer ← HeadPointer
               HeadPointer ← NewNodePointer
         ELSE
            // link new node between previous node and next node
            NodeArray[NewNodePointer].Pointer ← NextPOinter
            NodeArray[PreviousPointer].Pointer ← NewNodePointer
         END IF
   ENDIF
END PROCEDURE
```