

---

**COMPUTER SCIENCE**

**9608/41**

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<sup>®</sup>, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

Question	Answer				Marks
1(a)	<b>Label</b>	<b>Op code</b>	<b>Operand</b>	<b>Comment</b>	<div> <div> <div>1</div> <div>1</div> <div>1 + 1</div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> </div> </div>
	START:	IN		// INPUT character	
		STO	CHAR	// store in CHAR	
		LDM	#65	// Initialise ACC (ASCII value for 'A' is 65)	
	LOOP:	OUT		// OUTPUT ACC	
		CMP	CHAR	// compare ACC with CHAR	
		JPE	ENDFOR	// if equal jump to end of FOR loop	
		INC	ACC	// increment ACC	
		JMP	LOOP	// jump to LOOP	
	ENDFOR:	END			
	CHAR:				
1(b)	START:	LDD	NUMBER		<div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> <div>1</div> </div>

Question	Answer	Marks
2(a)	<p>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></p> <pre> DECLARE Customer : ARRAY[0 : 199] OF CustomerRecord FOR x ← 0 TO 199     Customer[x].CustomerID ← 0 ENDFOR </pre> <p>1 1 1+1</p>	4
2(b)(i)	<pre> PROCEDURE InsertRecord(BYVAL NewCustomer : CustomerRecord)     TableFull ← FALSE     // generate hash value     Index ← Hash(NewCustomer.CustomerID)     Pointer ← Index    // take a copy of index      // find a free table element     WHILE Customer[Pointer].CustomerID &gt; 0         Pointer ← Pointer + 1         // wrap back to beginning of table if necessary         IF Pointer &gt; 199             THEN                 Pointer ← 0             ENDIF         // check if back to original index         IF Pointer = Index             THEN                 TableFull ← TRUE             ENDIF     ENDWHILE     IF NOT TableFull         THEN             Customer[Pointer] ← NewCustomer         ELSE             OUTPUT "Error"         ENDIF     ENDPROCEDURE </pre> <p>1 1 1 1 1 1 1 1</p>	9

Question	Answer	Marks
2(b)(ii)	<pre> FUNCTION SearchHashTable(BYVAL SearchID : INTEGER) RETURNS INTEGER     // generate hash value     Index ← Hash(SearchID)     // check each record from index until found or not there     WHILE (Customer[Index].CustomerID &lt;&gt; SearchID)         AND (Customer[Index].CustomerID &gt; 0)         Index ← Index + 1     // wrap if necessary     IF Index &gt; 199         THEN             Index ← 0         ENDIF     ENDWHILE     // has customer ID been found?      IF Customer[Index].CustomerID = SearchID         THEN             RETURN Index         ELSE             RETURN -1         ENDIF     ENDFUNCTION </pre>	<p><b>9</b></p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p>
2(b)(iii)	A record out of place may not be found	<b>1</b>

Question	Answer	Marks
3	<pre> FUNCTION Find(BYVAL Name : STRING,               BYVAL Start : INTEGER,               BYVAL Finish : INTEGER) RETURNS INTEGER // base case IF <b>Finish &lt; Start</b>     THEN         RETURN -1     ELSE         Middle ← <b>(Start + Finish) DIV 2</b>         IF <b>NameList[Middle] = Name</b>             THEN                 RETURN <b>Middle</b>             ELSE // general case                 IF SearchItem &gt; <b>NameList[Middle]</b>                     THEN                         <b>Find(Name, Middle + 1, Finish)</b>                     ELSE                         <b>Find(Name, Start, Middle - 1)</b>                 ENDIF             ENDIF         ENDIF     ENDIF ENDFUNCTION </pre>	<p><b>7</b></p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p>

Question	Answer	Marks
4(a)(i)	containment/aggregation	<b>1</b>
4(a)(ii)	 <pre> classDiagram     class LinkedList {         &lt;     }     class Node {         &lt;     }     LinkedList "1" *-- "0..*" Node </pre> <p>1 mark for the two classes (in boxes) and connection with correct end point 1 mark for 0 ..* 0</p>	<b>Max 2</b>

Question	Answer	Marks
4(b)	<p><b>mark as follows:</b></p> <ul style="list-style-type: none"> <li>• Class heading and ending</li> <li>• Constructor heading and ending</li> <li>• Parameters in constructor heading</li> <li>• Declaration of (private) attributes : Pointer, Data</li> <li>• Assignment of parameters to Pointer and Data</li> </ul> <p><b>Python Example</b></p> <pre> class Node:     def __init__(self, D, P):         self.__Data = D         self.__Pointer = P         return </pre> <p><b>Example Pascal</b></p> <pre> type     Node = class         private             Data : String;             Pointer : Integer;         public             constructor Create(D : string; P : integer);             procedure SetPointer(P : Integer);             procedure SetData(D : String);             function GetData() : String;             function GetPointer() : Integer;         end;     constructor Node.Create(D : string; P : integer);     begin         Data := D;         Pointer := P;     end; </pre> <p><b>Example VB.NET</b></p> <pre> Class Node     Private Data As String     Private Pointer As Integer     Public Sub New(ByVal D As String, ByVal P As Integer)         Data = D         Pointer = P     End Sub End Class </pre>	<p><b>5</b></p> <p>1 1 + 1 1 1</p> <p>1 1</p> <p>ignore</p> <p>1+1</p> <p>1</p> <p>1 1 1+1 1</p>
4(c)(i)	A pointer that doesn't point to any data/node/address	<b>1</b>

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)	<p><b>mark as follows:</b></p> <ul style="list-style-type: none"> <li>• Class and constructor heading and ending</li> <li>• Declare private attributes (HeadPointer, FreeListPointer, NodeArray)</li> <li>• Initialise HeadPointer to null</li> <li>• Initialise FreeListPointer to 0</li> <li>• Looping 8 times ...</li> <li>• Creating empty node in NodeArray</li> <li>• Use .SetPointer method to point each new node to next node</li> <li>• Set last node pointer to null pointer</li> </ul> <p><b>Python Example</b></p> <pre> class LinkedList:     def __init__(self):         self.__HeadPointer = - 1         self.__FreeListPointer = 0         self.__NodeArray = []         for i in range(8):             ThisNode = Node("", (i + 1))             self.__NodeArray.append(ThisNode)         self.__NodeArray[7].SetPointer(- 1) </pre> <p><b>Example Pascal</b></p> <pre> type LinkedList = class 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(); var i : integer; begin     HeadPointer := -1;     FreeList := 0;     for i := 0 To 7 do         NodeArray[i] := Node.Create('', (i + 1));     NodeArray[7].SetPointer(-1); end; </pre>	Max 7

Question	Answer	Marks
	<p><b>Example VB.NET</b></p> <pre> Class LinkedList     Private HeadPointer As Integer     Private FreeList As Integer     Private NodeArray(7) As Node      Public Sub New()         HeadPointer = -1         FreeList = 0         For i = 0 To 7             NodeArray(i) = New Node("", (i + 1))         Next         NodeArray(7).SetPointer(-1)     End Sub End Class </pre>	<p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p>
4(c)(iv)	<ul style="list-style-type: none"> <li>Creating instance of LinkedList assigned to contacts</li> </ul> <p><b>Python Example</b></p> <pre>contacts = LinkedList()</pre> <p><b>Pascal Example</b></p> <pre> var contacts : LinkedList;     contacts := LinkedList.Create; </pre> <p><b>VB.NET Example</b></p> <pre>Dim contacts As New LinkedList</pre>	1



Question	Answer	Marks
4(c)(v)	<p><b>mark as follows:</b></p> <ul style="list-style-type: none"> <li>• Start with HeadPointer</li> <li>• Output node data</li> <li>• Loop until null pointer</li> <li>• Following pointer to next node</li> <li>• Use of getter (ie GetData/GetPointer)</li> </ul> <p><b>Python Example</b></p> <pre>def OutputListToConsole(self) :     Pointer = self.__HeadPointer     while Pointer != -1 :         print(self.__NodeArray[Pointer].GetData())         Pointer = self.__NodeArray[Pointer].GetPointer()     print()     return</pre> <p><b>Pascal Example</b></p> <pre>procedure LinkedList.OutputListToConsole(); var Pointer : integer; begin     Pointer := HeadPointer;     while Pointer &lt;&gt; -1 do         begin             WriteLn(NodeArray[Pointer].GetData);             Pointer := NodeArray[Pointer].GetPointer;         end;     end;</pre> <p><b>VB.NET Example</b></p> <pre>Public Sub OutputListToConsole()     Dim Pointer As Integer     Pointer = HeadPointer     Do While Pointer &lt;&gt; -1         Console.WriteLine(NodeArray(Pointer).GetData)         Pointer = NodeArray(Pointer).GetPointer     Loop End Sub</pre>	<p><b>5</b></p> <p>1</p> <p>1</p> <p>1+1</p> <p>1</p> <p>1</p> <p>1</p> <p>1+1</p> <p>1</p> <p>1</p> <p>1</p> <p>1+1</p> <p>1</p>

Question	Answer	Marks
4(c)(vi)	<p><b>mark as follows:</b></p> <ul style="list-style-type: none"> <li>• Store free list pointer as NewNodePointer</li> <li>• Store new data item in free node</li> <li>• Adjust free pointer</li> <li>• F list is currently empty</li> <li>• Make the node the first node</li> <li>• Set pointer of this node to Null Pointer</li> <li>• Find insertion point</li> <li>• If previous pointer is Null pointer</li> <li>• Link this node to front of list</li> <li>• Link new node between Previous node and next node</li> </ul> <p><b>Python Example</b></p> <pre>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)</pre>	<b>Max 6</b>

Question	Answer	Marks
	<p><b>Pascal Example</b></p> <pre> 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;     end; end; </pre>	

Question	Answer	Marks
	<p><b>VB.NET Example</b></p> <pre> 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 is currently 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 </pre>	

Question	Answer	Marks
	<p>Pseudocode for reference:</p> <pre> 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 </pre>	