
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[®], Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

© IGCSE is a registered trademark.


This document consists of **13** printed pages.

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

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>4</p> <p>1 1 1+1</p>
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 > 0 Pointer ← Pointer + 1 // wrap back to beginning of table if necessary IF Pointer > 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>9</p> <p>1 1 1 1 1 1 1 1 1</p>

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 <> SearchID) AND (Customer[Index].CustomerID > 0) Index ← Index + 1 // wrap if necessary IF Index > 199 THEN Index ← 0 ENDF ENDWHILE // has customer ID been found? IF Customer[Index].CustomerID = SearchID THEN RETURN Index ELSE RETURN -1 ENDF ENDFUNCTION </pre>	<p style="text-align: right;">9</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p>
2(b)(iii)	A record out of place may not be found	1

Question	Answer	Marks
3	<pre> FUNCTION Find(BYVAL Name : STRING, BYVAL Start : INTEGER, BYVAL Finish : INTEGER) RETURNS INTEGER // base case IF Finish < Start THEN RETURN -1 ELSE Middle ← (Start + Finish) DIV 2 IF NameList[Middle] = Name THEN RETURN Middle ELSE // general case IF SearchItem > NameList[Middle] THEN Find(Name, Middle + 1, Finish) ELSE Find(Name, Start, Middle - 1) ENDIF ENDIF ENDIF ENDFUNCTION </pre>	<p>7</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	1
4(a)(ii)	 <pre> classDiagram class LinkedList class Node 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>	Max 2

Question	Answer	Marks
<p>4(b)</p>	<p>mark as follows:</p> <ul style="list-style-type: none"> • 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 <p>Python Example</p> <pre> class Node: def __init__(self, D, P): self.__Data = D self.__Pointer = P return </pre> <p>Example Pascal</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>Example VB.NET</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>5</p> <p>1 1 + 1 1 1</p> <p>1 1</p> <p>ignore</p> <p>1+1 1</p> <p>1 1 1+1 1</p>
<p>4(c)(i)</p>	<p>A pointer that doesn't point to any data/node/address</p>	<p>1</p>

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>mark as follows:</p> <ul style="list-style-type: none"> • 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 <p>Python Example</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>Example Pascal</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>	<p>Max 7</p> <p>1 1</p>

Question	Answer	Marks
	<p>Example VB.NET</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> <p>1</p>
4(c)(iv)	<ul style="list-style-type: none"> • Creating instance of LinkedList assigned to contacts <p>Python Example</p> <pre>contacts = LinkedList()</pre> <p>Pascal Example</p> <pre>var contacts : LinkedList; contacts := LinkedList.Create;</pre> <p>VB.NET Example</p> <pre>Dim contacts As New LinkedList</pre>	1

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

Question	Answer	Marks
4(c)(vi)	<p>mark as follows:</p> <ul style="list-style-type: none"> • 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 <p>Python Example</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>	Max 6

Question	Answer	Marks
	<p>Pascal Example</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>VB.NET Example</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>	