Question	Answer	Marks
8(a)	**One** mark for each correct marking point **(Max 2)** • The initial order of the data • The number of data items to be sorted • The efficiency of the sorting algorithm	**2**

Question	Answer	Marks

8(b)	**One** mark for each marking point **(max 6)**
	
	MP1 Use of FOR loop to cycle through the whole year group
	MP2 Temporary storage of the score being 'inserted'
	MP3 Temporary storage of the corresponding name elements
	MP4 Use of WHILE loop with correct exit clause
	MP5 Moving of all three elements of data to next array elements
	MP6 Correct updating of counter variable
	MP7 Final insertion of all three data elements
	
	Example algorithm
	`YearSize ← 249`
	`FOR Student ← 2 to YearSize`
	`  Temp1 ← Score[Student]`
	`  Temp2 ← Name[Student,1]`
	`  Temp3 ← Name[Student,2]`
	`  Counter ← Student`
	`  WHILE Counter > 1 AND Score[Counter - 1] < Temp1`
	`  Score[Counter] ← Score[Counter - 1]`
	`  Name[Counter,1] ← Name[Counter - 1,1]`
	`  Name[Counter,2] ← Name[Counter - 1,2]`
	`  Counter ← Counter - 1`
	`  ENDWHILE`
	`  Score[Counter] ← Temp1`
	`  Name[Counter,1] ← Temp2`
	`  Name[Counter,2] ← Temp3`
	`NEXT Student`
		6

Question	Answer	Marks
10(a)	**One** mark for each correct marking point **(Max 3)** • Must have a base case/stopping condition • Must have a general case • … which calls itself (recursively) // Defined in terms of itself • … which changes its state and moves towards the base case Unwinding can occur once the base case is reached.	**3**
10(b)	**One** mark for each correct marking point **(Max 3)** • A stack is a LIFO data structure • Each recursive call is pushed onto the stack • …. and is then popped as the function ends • Enables backtracking/unwinding … to maintain the required order.	**3**
10(c)	**One** mark for each marking point **(Max 2)** • Linked List • Queue Binary Tree	**2**

10(d)	**One** mark for each marking point **(Max 5)**
	
	• Checking if stack is full / empty using `IF … THEN … (ELSE) … ENDIF`
	• … correctly using `StackFull()` function
	• **`RETURN`** suitable message if stack is full
	• **`RETURN`** message if space available on stack
	• Incrementing `TopOfStack` pointer if space available
	• Assigning new data using correct `NewInteger` variable
	• … to correct the array element in `ArrayStack[]` array.
	
	Example algorithm
	`FUNCTION AddInteger(NewInteger : INTEGER) RETURNS STRING`
	`  IF StackFull() THEN`
	`  RETURN "The stack is full"`
	`  ELSE`
	`  TopOfStack ← TopOfStack + 1`
	`  ArrayStack[TopOfStack] ← NewInteger`
	`  RETURN "Item added"`
	`  ENDIF`
	`ENDFUNCTION`
		5

Question	Answer	Marks

12(a)	**One** mark for each point (**Max 6**)
	
	• Initialisation of upper bound
	• Test if upper bound is less than lower bound
	• Re-setting of mid value if current value is lower than the target
	• Re-setting of mid value if current value is higher than the target
	• Finding the value
	• Correct termination of loop
	
	`Lower ← 0`
	`Upper ← 99`
	`Mid ← 0`
	`Exit ← FALSE`
	`OUTPUT "Enter the name to be found "`
	`INPUT Target`
	`REPEAT`
	`  IF Upper < Lower THEN`
	`  OUTPUT Target, " does not exist"`
	`  Exit ← TRUE`
	`  ENDIF`
	`  Mid ← Lower + (Upper - Lower + 1) DIV 2`
	`  IF Names[Mid] < Target THEN`
	`  Lower ← Mid + 1`
	`  ENDIF`
	`  IF Names[Mid] > Target THEN`
	`  Upper ← Mid - 1`
	`  ENDIF`
	`  IF Names[Mid] = Target THEN`
	`  OUTPUT Target, " was found at location ", Mid`
	`  Exit ← TRUE`
	`  ENDIF`
	`UNTIL Exit // UNTIL Exit = TRUE`
		6

12(b)(i)	O(n)	**1**

Question	Answer	Marks
12(b)(ii)	**One** mark for each point (**Max 2**) • O(log n) is a time complexity that uses logarithmic time. • The time taken goes up linearly as the number of items rises exponentially • O(log n) is the worst case scenario (time complexity for a binary search).	**2**