3.5 Sorting algorithms Vectors Mark Scheme

Q1.

(a) **Mark is for AO1 (knowledge)**

`n^2 // O(n^2);`
A. other ways of indicating `n^2` e.g. `n^2`
A. `On^2`

1

(b) **Marks are for AO1 (understanding)**

In each pass through the list `n` items will be examined;
There will be (at most) `n` passes through the list;

2

[3]

Q2.

(a) **All marks AO2 (analyse)**

1 mark: The arrow should be pointing towards the base class;
1 mark: There is no class called `Monster` / / it should say `Enemy`, not `Monster;`

2

(b) **Mark is for AO2 (apply)**

VB.Net
```
Dim MyGame As New Game(False) / /
Dim MyGame As New Game(True) / /
Private Player As New Character / /
Private Cavern As New Grid(NSDistance, WEDistance) / /
Private Monster As New Enemy / /
Private Flask As New Item / /
Private Trap1 As New Trap / /
Private Trap2 As New Trap;
```

R If any additional code
R If spelt incorrectly
I Case

1

(c) **Mark is for AO2 (apply)**

VB.Net
`CavernState;`

R If any additional code
R If spelt incorrectly
I Case

1

(d) **Mark is for AO2 (apply)**
`Trap / / Character / / Enemy;`
A `SleepyEnemy`
R If any additional code

R If spelt incorrectly
I Case

1

(e) **Mark is for AO2 (apply)**

```
Choice / / NoOfCellsEast / / NoOfCellsSouth / / Count / / NSDistance
/ / WEDistance / / Count1 / / Count2;
```

R If any additional code
R If spelt incorrectly
I Case

1

(f) **Mark is for AO2 (apply)**

```
Game;
```

R If any additional code
R If spelt incorrectly
I Case

1

(g) **Mark is for AO2 (analyse)**

So that a position of (0,0) is rejected / / so that the item can't be in the player's starting position;

1

(h) **Marks are for AO1 (understanding)**

Makes the program code easier to understand;
Makes it easier to update the program;
Makes it easier to change the size of the cavern (in the game);
Max 2 points from the list above

2

(i) **Marks are for AO2 (analyse)**
1 mark: Create a new object (`Trap3`) of class `Trap;`
1 mark: Change the (3rd) `If` statement in the `PlayGame` subroutine by adding conditions to check if the player is in the same cell as `Trap3` and that `Trap3` has not been triggered already;

2

[12]

Q3.

(a) Sort the list of numbers / / Sort `L`;

1

(b) The initial situation;

1

(c) Ownership;
Resources;
Constraints;

MAX 2

(d) `FOR Count2 ← 1 TO (MAX - 1);`
A Any answer where meaning is clear

1

(e) `L[Count2] ← L[Count2 + 1];`

A Any answer where meaning is clear

1

(f) `L[Count2 + 1] ← Temp;`

A Any answer where meaning is clear

1

(g) 63;

1

(h) Set `SwapMade` to have a value of `False` before the inner loop starts;
If a swap is made then set `SwapMade` to `True`;
Change the outer loop so that it keeps on repeating until `SwapMade` equals `False`;

Note: if neither of the first 2 mark points have been awarded 1 mark should be awarded for the idea of creating a flag / Boolean variable

Alternative answer
Set `NoMoreSwaps` to have a value of `True` before the inner loop starts;
If a swap is made then set `NoMoreSwaps` to `False`;
Change the outer loop so that it keeps on repeating until `NoMoreSwaps` equals `True`;

Note: if neither of the first 2 mark points have been awarded 1 mark should be awarded for the idea of creating a flag / Boolean variable

Alternative answer
Set `NoOfSwaps` to have a value of `0` before the inner loop starts;
If a swap is made then increment `NoMoreSwaps`;
Change the outer loop so that it keeps on repeating until `NoMoreSwaps` equals `0;`

Note: if neither of the first 2 mark points have been awarded 1 mark should be awarded for the idea of creating a counter variable

A Any sensible identifier
A No identifier specified
A Alternative sensible data type
A Pseudo-code answers

3

[11]

Q4.

(a)

Algorithm Name	Requires Sorted List? (Tick one box)
Binary search	✓
Linear search	

1 mark for having a tick in the "Binary search" row.

A alternative indicators for tick eg "Yes"
A a tick for "Binary search" and a cross for "Linear search"
R answers where two ticks have been used.

1

(b)

				List			
List Length	Outer Pointer	Current Value	Inner Pointer	[1] 9	[2] 8	[3] 5	[4] 6
4	2	8	1		9		
			0	8			
	3	5	2			9	
			1		8		
			0	5			
	4	6	3				9
			2			8	
			1		6		

Award **1 mark** for each of the highlighted rectangles which has the correct values written in it in the unshaded cells.
Accept responses in which correct values are unnecessarily written out again.
Do not award a mark for any rectangle which has an incorrect value written in it.

3

(c) The value being moved / CurrentValue / 6 does not need to be put at the start of the list / / should be inserted at position 2 not position 1;
Because the second condition (in the While statement) is not satisfied;
MAX 1

1

(d)

Order of Time Complexity	Tick one box
O(n)	
$O(n^2)$	✓
$O(2^n)$	

A alternative indicators instead of a tick eg a cross, Y, Yes
R responses in which more than one box is ticked

1

(e) Insertion sort;

A Insert sort

1

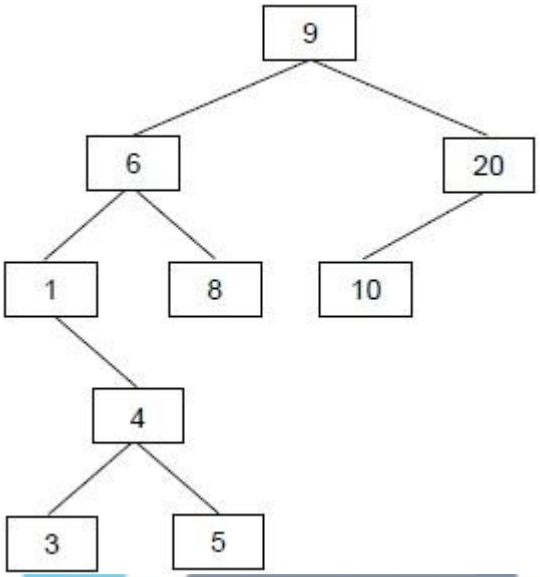
(f) (i) 9, 6, 8;
Must be in the order above. Can be separated by any character or a space

1

(ii) 9, 20,10;
Must be in the order above. Can be separated by any character or a space.

1

(g)



1 mark for inserting number 4 in the correct place
1 mark for inserting both numbers 3 and 5 in the correct place relative to 4
MAX 1 if any numbers added in the wrong place / any extra numbers added

2

[11]

Q5.

(a) 18, 23, 21, 36, 40, 45, 58, 59

Mark as follows:
18 in the first place;
23 and 21 in correct order and in the second and third places;
21 and 36 in the correct order and in the third and fourth places;
40, 45, 58 and 59 in the correct order and in the last four places;
A Table 3 instead of Table 2 as long as the bottom cell of each of the scores column is correct (**I.** any working out)

4

(b) Bubble sort;
NE sort

1

[5]

Q6.

(a)

Ptr	Temp	List									
		[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
		43	25	37	81	18	70	64	96	52	4
1	43	25	43								
2	43		37	43							
3											
4	81				18	81					
5	81					70	81				
6	81						64	81			
7											
8	96								52	96	
9	96									4	96
10											

Ignore Ptr & Temp column
1 mark for each of rows 1, 2, 4, 5, 6, 8, 9
(Final list 25, 37, 43, 18, 70, 64, 81, 52, 4, 96)

7

(b) Control will pass to the instruction after Endwhile;
/the instruction/command/statement after Endwhile will be executed;
Program will exit while-block; loop stops;
A algorithim stops; **R** program stops;

Max 1

(c) (i) 25;

If part (a) not fully correct allow follow through: or lower of [1] & [2]

3

(ii) 81;

Only allow follow through mark if the list at the end of part(a) is still a partially sorted list

(iii) 96;

Must be 96 in all cases

[11]

Q7.

(a) 11, 17, 9,21,15,23;

(2 if all right, 1 if 4 of 6)
If > misinterpreted, follow through for 1 mark

2

(b) A bubble sort;

1

(c) To detect when all the numbers have been sorted
Efficiency (to stop procedure repeating unnecessarily);
R to detect when numbers have switched

1

[4]

Q8.

(a) See trace table

10

(b) Insertion sort

1

(c) Time taken (1)
To move many items / to make space for one insertion.(1)

2

[13]

Trace table

Comment	count	rp	max	cp	temp	numbers			
						1	2	3	
Global values on call			3			13	25	24	
rp:=1		1							1 mark for assigning and incrementing rp and assigning cp
repeat									
rp:=rp+1		2							
cp:=1				1					
while rp>cp do									
if numbers[rp] > numbers[cp] then									
temp:= numbers[rp]					25				1 mark for temp
for count:=rp to cp+1 step- 1	2								1 mark for count starting from 2 and numbers [2] correct (no need to show count dropping to 1)

numbers[count]:= numbers[count-1]							13		
endfor	1								
numbers[cp]:=temp						25			1 mark for copying temp to numbers[1]
endif									
cp:=cp+1				2					1 mark for incrementing cp (carry forward error)
endwhile									
until rp=max									
rp:=rp+1		3							1 mark for rp incremented
cp:=1				1					And cp assigned 1
While rp>cp do									
if numbers[rp] > numbers[cp] then									
endif									
cp=cp+1				2					1 mark for cp incremented

endwhile									
if numbers[rp] > numbers[cp] then									
temp:= numbers[rp]					24				1 mark for numbers[3] copied to temp
for count:=rp to cp+1 step-1	3								1 mark for count starting from 3 and numbers [3]
numbers[count]:= numbers[count-1]								13	correct (no need to show count dropping to 2)
endfor	2								

numbers[cp]:=temp							24		1 mark for numbers[2] assigned 24 And cp incremented
endif									
endif									
cp:=cp+1				3					
endwhile									
until rp=max									

Examiner reports

Q1.

This was the Section A question that students found hardest, with very few getting full marks. Not many students were able to identify the time complexity of either merge sort or (to a lesser extent) bubble sort and a significant number of students thought that the binary search and/or the post-order tree traversal would not be used to solve tractable problems.

When students could state the time complexity of the bubble sort algorithm they were rarely able to clearly explain why $O(n^2)$ was the correct answer.

Q3.

Students found this question considerably harder than the other questions in Section A. Most students could state at least one other component of a well-defined problem for part (c) but for part (b) a lot of answers described the given of this particular problem instead of defining what was meant by the given of a problem. 6 was the most common wrong answer for part (g) as a number of students assumed that the sort would be in ascending order. Few students scored good marks on part (h), descriptions were often poorly-written or demonstrated no understanding of the bubble sort algorithm.

Q4.

(a) The overwhelming majority of students were able to correctly identify that it was the binary search algorithm that required the list to be sorted for this part.

(b) The trace for this part was also well completed with about three quarters of students getting some marks and well over half getting full marks.

(c) For this part, around half of the candidates were correctly able to explain that the value of InnerPointer did not decrease to zero because either the second while loop condition was not

(d) For this part, about two thirds of students correctly identified that the algorithm that they had traced was of time complexity $O(n^2)$.

(e) This part was poorly answered, with only about one third of students correctly identifying that the algorithm they had traced was an insertion sort. Bubble sort was a far more common but incorrect response.

(f) Parts (i), (ii) were all well answered. The most common error in both parts of was to perform a traversal of the tree instead of using it as a binary search tree.

(g) This part was all well answered. The most common error in both parts of was to perform a traversal of the tree instead of using it as a binary search tree.

Q5.

Many candidates were able to identify that the algorithm in the figure was a bubble sort; a significant number of these were also able to complete Table 2 correctly. Some candidates worked out (or guessed) that the algorithm was a bubble sort and then completed Table 2 by filling out all the Scores in the correct order instead of doing what the question asked which was to complete one pass through the outer loop. Candidates must ensure that they read questions carefully and answer what has been asked.

Q6.

(a) There was a definite improvement in the dry running of a piece of pseudo-code. Many candidates were able to correctly complete the table to gain the 7 marks. Some candidates instantly recognised the bubble sort and decided to skip the dry run and write the final values on the bottom line (the mark scheme penalised this severely). Others assumed that it was a complete bubble sort and simply wrote down the list of numbers in ascending order. However, a significant number of candidates still did not seem to be prepared for dry-running an algorithm. The grid was left blank or was completely filled in with totally irrelevant numbers by such candidates.

(b) This was poorly answered with many candidates stating that the program stopped or that 'nothing will happen'. A few candidates gained credit by stating that control will pass to the instruction after EndWhile.

(c) Those candidates who answered (a) correctly obtained at least 2 marks for this part. Even good candidates often did not spot that another run through the algorithm would put 81 into the ninth element of List.

Q7.

This was on the bubble sort algorithm. Space was left here for candidates to work through the algorithm if they wished. Many candidates could identify the algorithm as a bubble sort, but failed to work through it correctly to gain marks in part (a). Many could see what the flag did, but could not explain that its purpose in the algorithm was to detect when all the numbers had been sorted so that the procedure was not repeated unnecessarily.

Q8.

In part (a) the majority of candidates managed to trace the algorithm and many obtained all ten marks. Some of the trace tables were very well laid out and easy to follow. Candidates should not cross out values when they change during the trace. Marks are obtained for showing the changes and crossing out values makes it difficult to read what was there. Some candidates are unwilling to use a new line for each stage of the trace, filling in the next available space under the appropriate heading. The resulting table gives no indication of sequence. Few candidates made any use of the comment column. A few candidates decided that the array contents would be sorted into ascending order and tried to make this happen with no regard for the algorithm. There were also candidates with no idea of how to trace the changes in the array elements.

In part (b) while many candidates correctly identified the insertion sort, a large minority gave bubble sort. This is surprising as the bubble sort is not on the syllabus.

In part (c) some candidates gained a mark for saying that the sort would take too much time but few understood that the method is inefficient because of the very large number of comparisons and exchanges needed.