EXAM PAPERS PRACTICE

3.5 Sorting algorithms

Name: ______________________

Class: ______________________

Date: ______________________

Time:	**94 minutes**
Marks:	**70 marks**
Comments:	

Q1.

(a) State the time complexity for the bubble sort algorithm in terms of n, where n is the number of items in the list to be sorted.

(1)

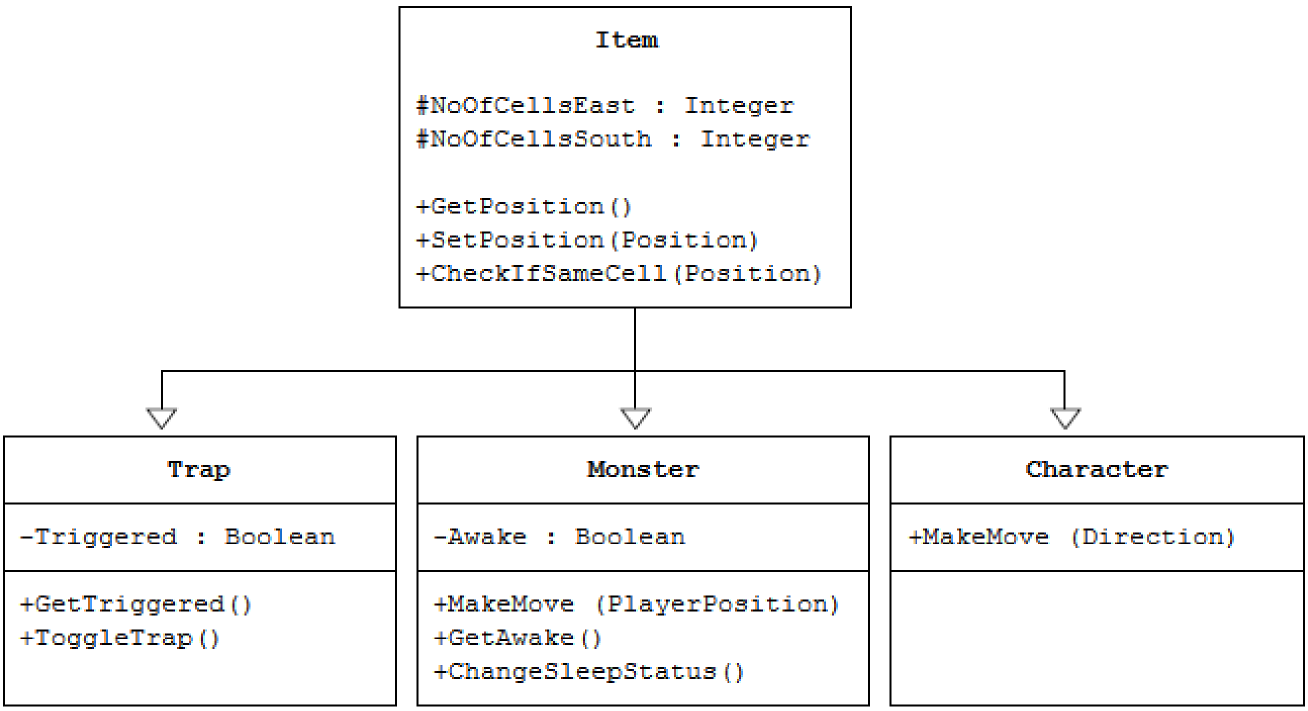
(b) Explain why the bubble sort algorithm has the time complexity stated in your answer to part **(a)**.

(2)

(Total 3 marks)

Q2.

The class diagram below is an attempt to represent the relationships between some of the classes in the MONSTER! Game.



(a) Explain what errors have been made in the class diagram.

(2)

(b) Give an example of instantiation from the **Skeleton Program**.

(1)

(c) State the name of an identifier for an array variable.

(1)

(d) State the name of an identifier for a subclass.

(1)

(e) State the name of an identifier for a variable that is used to store a whole number.

(1)

(f) State the name of an identifier for a class that uses composition.

(1)

(g) Look at the `GetNewRandomPosition` subroutine in the `Game` class in the **Skeleton Program**.

Explain why the generation of a random position needs to be inside a repetition structure.

(1)

(h) Look at the `Game` class in the **Skeleton Program**.

Why has a named constant been used instead of the numeric value `5`?

(2)

(i) Describe the changes that would need to be made to the `Game` class to add a third trap to the cavern. The third trap should have exactly the same functionality as the other two traps. You do **not** need to describe the changes that would need to be made to the `SetUpGame` subroutine.

(2)
(Total 12 marks)

Q3.

Figure 1 shows ten numbers stored in an array `L`.

Figure 1

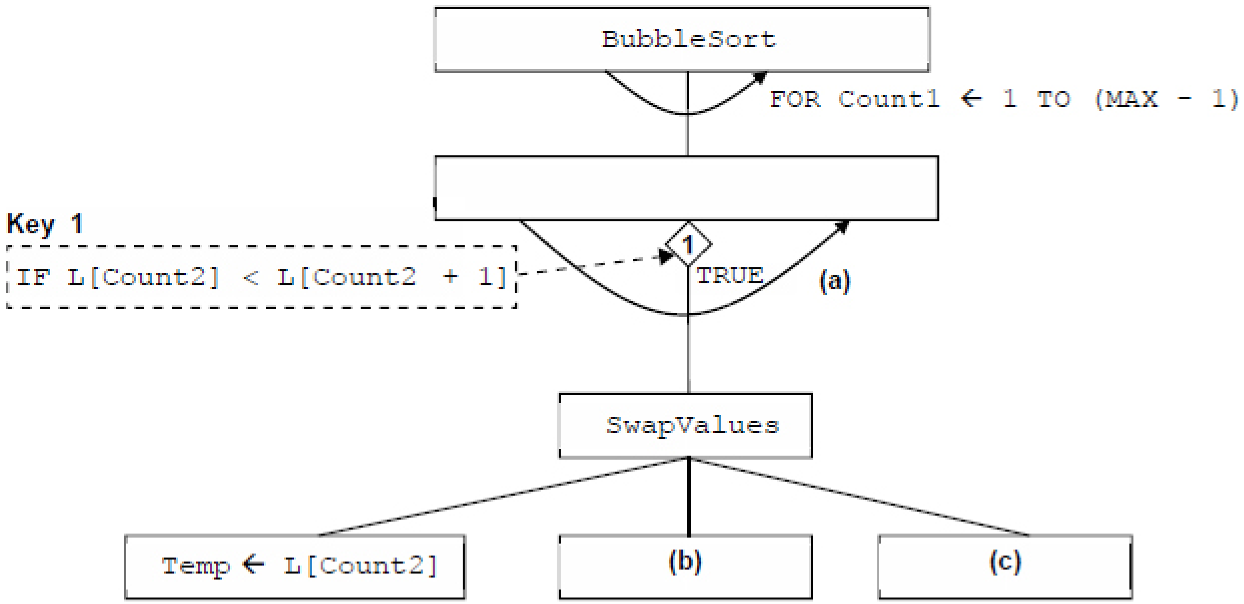
L									
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
34	8	6	35	27	35	63	56	16	24

The numbers in `L` are to be sorted.

Figure 2 shows an **incomplete** structure chart that has been created while developing a solution to the problem of sorting the numbers in `L`.

The constant `MAX` has been used to represent the size of the array.

Figure 2



(a) Describe the **goal** of this problem.

____

____

(1)

(b) What is meant by the **given** of a problem?

____

____

(1)

(c) The given and the goal are two components of a well-defined problem. State **two** other components of a well-defined problem.

____

____

____

____

(2)

(d) How should the curved arrow **(a)** in **Figure 2** be labelled?

____

____

(1)

(e) What should be written in box **(b)** in **Figure 2**?

____

____

(1)

(f) What should be written in box **(c)** in **Figure 2**?

____

____

(1)

A **new** Bubble Sort routine is developed using the structure chart shown in **Figure 2**.

(g) What value will be in `L[1]` when **this** Bubble Sort routine has finished executing on the array `L` shown in **Figure 1**?

____

____

(1)

(h) A Bubble Sort routine, based on the structure chart in **Figure 2**, always completes `MAX – 1` passes through the array. Often, this number of passes is not required, as the contents of the array will be sorted after fewer passes have been made.

If a pass is made through the array during which no swaps need to be made then the array has been sorted.

Describe the changes that need to be made to the Bubble Sort routine so that it will only complete the minimum number of passes through the array that are needed to fully sort the contents of the array.

____

____

____

____

____

____

(3)
(Total 11 marks)

Q4.

A computer program stores a list of integers in an array named `List`. The numbers in the array are to be sorted into ascending order so that a particular efficient search algorithm can be used to search for a number.

(a) One of the search algorithms in **Table 1** can only be used successfully on a sorted list.

Place **one** tick next to the name of the algorithm that requires a list to be sorted.

Table 1

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

(1)

(b) The pseudo-code for a standard algorithm that can be used to sort the data in the array `List` into order is shown in **Figure 1**. The variable `ListLength` stores a count of the number of items in the array `List`.

Array indexing starts at 1.

Figure 1

```
For OuterPointer ← 2 To ListLength
  CurrentValue ← List[OuterPointer]
  InnerPointer ← OuterPointer - 1
  While InnerPointer > 0 And
        List[InnerPointer] > CurrentValue Do
    List[InnerPointer + 1] ← List[InnerPointer]
    InnerPointer ← InnerPointer - 1
  EndWhile
  List[InnerPointer + 1] ← CurrentValue
EndFor
```

Complete the empty (unshaded) cells in the trace table (**Table 2**) for an execution of the algorithm in **Figure 1** when the array `List` contains the values `9, 8, 5` and `6` in that order.

Table 2

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

			2				
			1				

(3)

(c) In the trace table (**Table 2**), when the variable `OuterPointer` contains the value `2` and then `3`, the value of the variable `InnerPointer` decreases to `0`. When `OuterPointer` contains `4`, `InnerPointer` stops decreasing when it reaches the number `1`.

Explain why `InnerPointer` does not decrease to `0` when `OuterPointer` contains `4`.

____

____

____

____

(1)

(d) Tick **one** box in **Table 3** to indicate the correct Order of **Time** Complexity of the standard algorithm in **Figure 1**.

Table 3

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

(1)

(e) State the name of the standard algorithm that is represented by the pseudo-code in **Figure 1**.

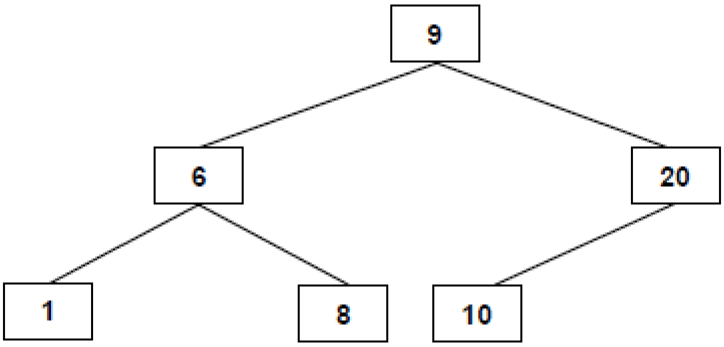
____

(1)

(f) Instead of storing a list of numbers in an array as in (b), the numbers could be stored in a binary search tree. This would also enable efficient searching.

The numbers 9, 6, 1, 8, 20 and 10 are put into a binary search tree in that order. **Figure 2** shows this binary search tree.

Figure 2



(i) A search of the binary tree is performed for the number `8`.

List the numbers, in the order that they would be checked, for the search to determine that the number `8` **is present** in the tree.

____

(1)

(ii) A search of the binary tree is performed for the number 11.

List the numbers, in the order that they would be checked, for the search to determine that the number `11` **is not present** in the tree.

____

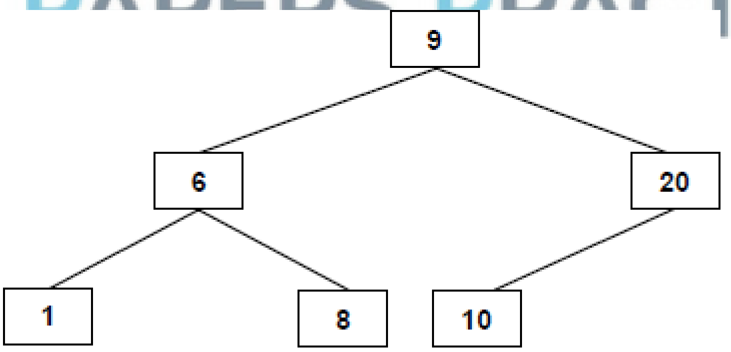
(1)

(g) The numbers 4, 5 and 3 are to be added into the binary search tree, in that order.

Figure 3 below is an identical copy of **Figure 2**.

Complete **Figure 3** below to show the binary search tree from **Figure 2** after the extra numbers have been added into it.

Figure 3



(2)

(Total 11 marks)

Q5.

The contents of an array `Scores` are shown in the table below.

A pseudo code representation of an algorithm is given below the table.

`Scores`

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
18	23	36	21	58	40	45	59

```
Max ← 8
FOR Count1 ← 1 TO (Max – 1) DO
  FOR Count2 ← 1 TO (Max – 1) DO
    IF Scores[Count2] > Scores[Count2 + 1]
      THEN
        Temp ← Scores[Count2]
        Scores[Count2] ← Scores[Count2 + 1]
        Scores[Count2 + 1] ← Temp
    ENDIF
  ENDFOR
ENDFOR
```

(a) **One pass** is made through the outer loop of the algorithm in the diagram above.

Complete **Table 1** to show the changed contents of the array `Scores` after this single pass. You may use **Table 2** to help you work out your answer, though you not required to use **Table 2**.

Table 1

Scores							
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

Table 2

Max	Count1	Count2	Temp	Scores							
				[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
				18	23	36	21	58	40	45	59

(4)

(b) What is the name of the standard algorithm shown in the pseudo code above?

____

(1)
(Total 5 marks)

Q6.

The algorithm below re-arranges numbers stored in a one-dimensional array called **List**. **Ptr** is an integer variable used as an index (subscript) which identifies elements within **List**. **Temp** is a variable, which is used as a temporary store for numbers from **List**.

```
Ptr ← I
While Ptr < 10 Do
  If List [Ptr] > List [Ptr+ 1] Then

     Temp ← List [Ptr]

     List [Ptr] ← List [Ptr+l]

     List [Ptr+l] ← Temp
  Endif

  Ptr ← Ptr+ 1
Endwhile
```

(a) Dry-run the algorithm by completing the table below,

It is only necessary to show those numbers which change at a particular step.

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

(7)

(b) What will happen when **Ptr**= 10?

____

(1)

(c) If the whole algorithm is now applied to this rearranged list, what will be the values of:

(i) List[1] ____

(ii) List[9] ____

(iii) List[10]? ____

(3)
(Total 11 marks)

Q7.

A procedure to process an array of numbers is defined as follows.

```
Procedure P(Number)
  Repeat

      X ← StartofArray

      Flag ← False
      Repeat
        If Number(X) > Number (X+ 1)
           Then
             Begin

               Temp ← Number(X)

               Number (X) ← Number (X+ 1)

               Number(X+I) ← Temp

               Flag ← True
             End

           X ← X+l
        Until EndofArray
      Until Flag = False
Endproc
```


The array number, containing 17, 11, 21, 9, 23, 15, is to be processed by this procedure.

(a) List the array after the outer Repeat loop has been executed once.

____

____

____

____

(2)

(b) What algorithm does the procedure P describe?

(1)

(c) What is the purpose of Flag in this procedure?

(1)
(Total 4 marks)

Q8.

The algorithm below shows a procedure called sort.

```
//numbers is a global array of integers
// max is a global integer holding the number of values to be sorted

procedure sort

    integer: cp, rp, temp, count
    rp := 1

    repeat

        rp := rp+1
        cp := 1
        while rp > cp do
            if numbers[rp] > numbers[cp] then
                temp := numbers[rp]
                for count := rp to cp + 1 step - 1
                    numbers[count] := numbers[count - 1]
                endfor
                numbers[cp] := temp
            endif
        cp := cp+1
        endwhile
    until rp = max

endproc
```

(a) Using the column headings shown below, trace the algorithm for the procedure *sort* when the array *numbers* contains the values 13, 25, 24 and max = 3.

Comment	Count	rp	max	cp	temp	numbers		
						1	2	3
Global values on call			3			13	25	24

(10)

(b) Name the sort method used in the algorithm above.

____

(1)

(c) Why would this method be inefficient if the array *numbers* contained 500 values?

____

____

____

____

(2)

(Total 13 marks)