Question			Marks
1	**1**	**Mark is for AO2 (apply)** 4! // 4x3x2x1 // 4x3x2 // 24;	**1**
1	**2**	**Mark is for AO2 (apply)** n! // factorial of n; **A.** 1 * 2 *… * n-1 * n (or similar)	**1**
1	**3**	**Mark is for AO2 (analyse)** The string could contain more than one occurrence of a character; Each athlete is unique, each character is not (guaranteed to be) unique; There are *n* characters in the string but not *n* distinct characters; Some of the anagrams could be duplicates; **Max 1**	**1**
1	**4**	**All marks for AO1 (knowledge)** A problem that can be solved; but not in a reasonable amount of time as the problem size increases // but has an exponential (or worse) time complexity // but there is no polynomial (or less) time solution;	**2**
1	**5**	**Mark is for AO1 (understanding)** One;	**1**
1	**6**	**One mark is for AO1 (knowledge) and one mark is for AO1 (understanding)** **AO1 knowledge** O(n); **Max 1 for AO1 understanding** As the size of the list increases the time taken increases at the same rate; There is a loop that repeats n times;	**2**

Question			Marks
1	**7**	**One mark is for AO1 (knowledge) and one mark is for AO1 (understanding)** **AO1 knowledge** O(log n); **Max 1 for AO1 understanding** Each comparison halves the size of the list that has to be searched through; The time taken increases as the size of the list increases but by smaller and smaller amounts; If the size of the list doubles then the number of comparisons needed only increases by 1;	**2**

<table>
<tr><th colspan="2">Question</th><th></th><th>Marks</th></tr>
<tr><td>2</td><td></td><td>

All marks AO1 (knowledge)

Algorithm	Time Complexity
Binary tree search	O(log n)
Bubble sort	$O(n^2)$
Linear search	O(n)
Merge sort	O(n log n)

A. O(n x log n) **NE.** O(log n)

I. missing brackets
I. missing O

Mark as follows:

1 mark: 1st row correct
1 mark: 2nd row correct
1 mark: 3rd row correct

</td><td>3</td></tr>
</table>

<table>
<tr><th colspan="2">Question</th><th></th><th>Marks</th></tr>
<tr><td>03</td><td>1</td><td><b>All marks for AO1 (understanding)</b>
A root (<b>A.</b> start) node; <b>A.</b> there is a parent-child relationship between nodes
Each node has no more than two child nodes; <b>R.</b> has two child nodes</td><td>2</td></tr>
<tr><td>03</td><td>2</td><td><b>All marks are for AO2 (apply)</b>
<table>
<tr><td><b>1</b></td><td>0</td></tr>
<tr><td><b>2</b></td><td>-1</td></tr>
<tr><td><b>3</b></td><td>True</td></tr>
<tr><td><b>4</b></td><td>Current ← Tree[Current].Right</td></tr>
<tr><td><b>5</b></td><td>Current ← Tree[Current].Left</td></tr>
<tr><td><b>6</b></td><td>False</td></tr>
</table>
<b>Note for examiners:</b> answers are in pseudo-code so accept any reasonable representation (including use of string or integer values for rows 3 and 6).
<b>Mark as follows:</b>
<b>1 mark</b>: row one correct
<b>1 mark</b>: row two correct
<b>1 mark</b>: rows three and six correct
<b>1 mark</b>: rows four and five correct</td><td>4</td></tr>
<tr><td>03</td><td>3</td><td><b>Mark is for AO1 (knowledge)</b>
O(log$_2$n);
<b>I.</b> missing brackets
<b>I.</b> missing O
<b>I.</b> missing $_2$</td><td>1</td></tr>
<tr><td>03</td><td>4</td><td><b>Mark is for AO1 (understanding)</b>
Every comparison halves the size of the binary tree to look at; (<b>A.</b> every comparison halves the size of the tree)</td><td>1</td></tr>
</table>

03	**5**	**Mark is for AO1 (understanding)** It does not have exponential (or worse) time complexity; It has a polynomial (or better) time complexity solution; **A.** It can be solved in a reasonable amount of time regardless of the problem/input size; **NE.** can be solved in a reasonable amount of time **Max 1**	**1**
03	**6**	**All marks AO1 (understanding)** Rules/knowledge (about the problem domain); Can be used to find a good/approximate but (probably) not optimal solution to a problem; Can reduce the size of the search/problem space // changing some constraints in the problem; **Max 2**	**2**
03	**7**	**All marks AO1 (knowledge)** As the size of the input/problem increases; the amount of time taken remains the same;	**2**