Notes. When you are asked to compute the order of a time complexity function you need to give the **tightest** order. So, for example, while it is true that the function $f(n) = n + 1$ is $O(n^2)$, you should indicate that $f(n)$ is $O(n)$ and not that $f(n)$ is $O(n^2)$.

You might find these facts useful: In a proper binary tree with n nodes the number of leaves is $(n + 1)/2$ and the number of internal nodes is $(n - 1)/2$. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

Part 1: Multiple Choice

Circle **only ONE** answer.

Each multiple choice question is worth 3.5 marks.

1. Let $f(n)$, $g(n)$, and $h(n)$ be three functions with positive values for every $n \geq 0$. Assume that $f(n) < g(n)$, and $g(n) < h(n)$ for all $n \geq 0$. Which of the following statements must be true for **every** set of functions $f(n)$, $g(n)$, $h(n)$ as above?
(A) $f(n)$ is not $O(g(n))$
(B) $f(n)$ is $O(h(n))$
(C) $(f(n) + h(n))$ is $O(g(n))$
(D) $g(n)$ is not $O(f(n))$
(E) $(f(n) \times g(n))$ is $O(h(n))$

2. Two algorithms, P and Q, have time complexities $p(n)$ and $q(n)$, respectively. If $p(n)$ is $O(q(n))$ and $q(n)$ is $O(p(n))$, then which of the following statements is true?
(A) When executed on any computer, P and Q will have exactly the same running time.
(B) When executed on any computer, P and Q will have exactly the same running time, but only on large inputs.
(C) P could be faster than Q or Q could be faster than P.
(D) P is faster than Q for very large values of n.
(E) Q is faster than P for very large values of n.

3. Consider the following algorithm.

```
Algorithm foo(n)
Input: Integer value n

j ← 0
i ← 0
while i < n do {
    if j < i then j ← j + 1
    else {
        i ← i + 1
        j ← 0
    }
}
```

What is the time complexity of the algorithm?
(A) $O(1)$
(B) $O(n)$
(C) $O(n \log n)$
(D) $O(i \times n)$
(E) $O(n^2)$

4. What is the solution of the following recurrence equation?

   $f(0) = 5$
   $f(n) = f(n-1) + 2$

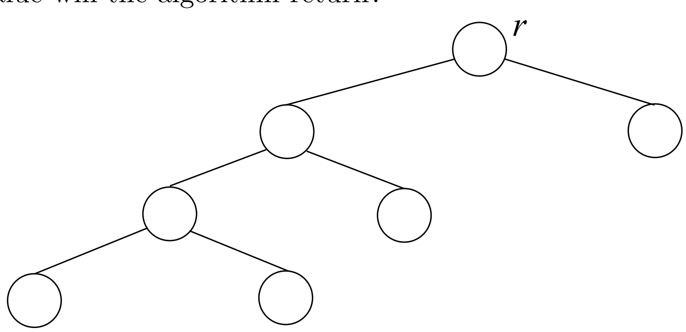
   (A) $f(n) = 5n + 2$
   (B) $f(n) = n + 5$
   (C) $f(n) = 2(n-1) + 5$
   (D) $f(n) = 2n + 5$
   (E) $f(n) = 2\frac{n-1}{2} + 5$

5. Which of the following initially empty hash tables has the smallest number of collisions when the keys 15, 11, 8, and 2 are inserted (in this order) in them? In all cases collisions are resolved using separate chaining.
   (A) A table of size 4 and hash function $h(k) = k \bmod 4$.
   (B) A table of size 5 and hash function $h(k) = k \bmod 5$.
   (C) A table of size 9 and hash function $h(k) = k \bmod 9$.
   (D) A table of size 7 and hash function $h(k) = k \bmod 7$.

6. Consider the following algorithm.

   Algorithm `eval`(r, k)
   Input: Root r of a proper binary tree and value $k \geq 0$.
   if r is a leaf **then return** k
   else return `eval`(r.left, $k+1$) + `eval`(r.right, $k+1$)

   Assume that above algorithm `eval` is performed over the following tree. The initial call is $i \leftarrow$ `eval`$(r, 0)$. What value will the algorithm return?
   (A) 20
   (B) 12
   (C) 9
   (D) 6
   (E) 5



7. A set of n keys: $\{k_1, \ldots, k_n\}$ is to be stored in an initially empty binary search tree. Which of the following statements is always true?
   (A) If k_i is the smallest key, then in every binary search tree storing the above set of keys, the left child of the node storing k_i is a leaf.
   (B) The resulting binary search tree has the same height, regardless of the order in which the keys are inserted in the tree.
   (C) A preorder traversal of the tree visits the keys in increasing order of value.
   (D) After inserting the keys, the key stored at the root of the tree is the same regardless of the order in which the keys are inserted.
   (E) None of the above statements is always true.

8. Let T be a proper binary tree with root r. Consider the following algorithm.

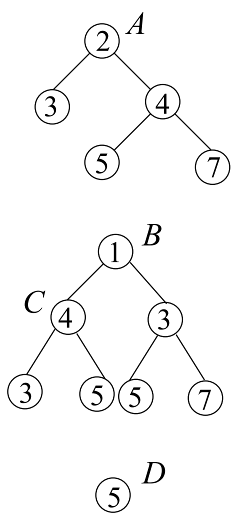
Algorithm `traverse`(r)
Input: Root r of a proper binary tree
if r is a leaf **then return** 1
else {
    $t \leftarrow$ `traverse`(r.left)
    $s \leftarrow$ `traverse`(r.right)
    return $s + t$
}

What does the algorithm do?
(A) It always returns the value 1.
(B) It computes the number of nodes in the tree.
(C) It computes the number of edges in the tree.
(D) It computes the height of the tree.
(E) It computes the number of leaves in the tree.

Part 2: Written Answers

9. [20 marks] A proper binary tree is a *min tree* if every node stores an integer key and for every internal node the key stored in it is smaller than the keys in its children. For example, the trees below with roots A and D are min trees but the tree with root B is not as node C has key 4 and its left child has key 3.

   Write in pseudocode an algorithm `minTree`(r) that receives as input the root r of a proper binary tree and it returns **true** if the tree is a min tree and **false** otherwise. Use r.key to denote the key of r, r.left and r.right to denote its children and assume that r.isLeaf has value **true** if r is a leaf and **false** otherwise.



Algorithm minTree(r)

10. [4 marks] Explain what the worst case for the algorithm is.

    [7.5 marks] Compute the time complexity of the above algorithm in the worst case as a function of the number n of nodes. You need to explain how you computed the time complexity.
    [0.5 marks] Compute the order ("bigh Oh") of the time complexity.

11. The following algorithm receives as input the root r of a proper binary tree with n nodes. Each node u stores a key, denoted as u.key. Let the height of the tree be h.

Algorithm `proc`(r)
In: Root r of a proper binary tree with n nodes

if r is a leaf **then return** r.key
else {
    $v_1 \leftarrow$ `proc`$(r.\text{left})$
    $v_2 \leftarrow$ `proc`$(r.\text{right})$
    return $r.key + v_1 + v_2$
}

[4 marks] How many calls to the algorithm are performed, including the initial call? Explain.

[6 marks] When executing this algorithm, what information is stored in each activation record? How much space is needed for each activation record (is it constant or does it depend on the number n of nodes)? Explain.

[8 marks] How much space does the execution stack need in the worst case? Explain.

12. Consider a hash table of size 7 with hash function $h(k) = k \bmod 7$. Draw the contents of the table after inserting, in the given order, the following values into the table:

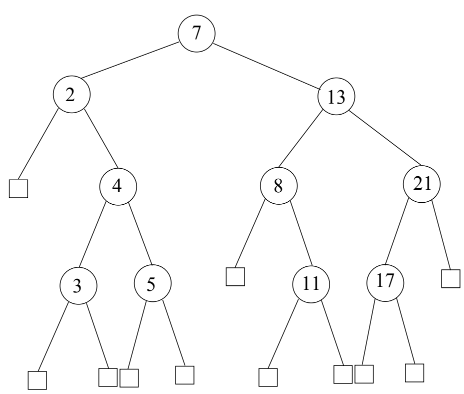
    9, 16, 27, 62, and 2:

    [4 marks] (a) when linear probing is used to resolve collisions

    [10 marks] (b) when double hashing with secondary hash function $h'(k) = 5 - (k \bmod 5)$ is used to resolve collisions.

	Linear probing		**Double hashing**
0		0	
1		1	
2		2	
3		3	
4		4	
5		5	
6		6	

13. [4 marks] Consider the following binary search tree. Insert the keys 9, 12, and 10, in that order, into the tree and show the resulting tree (you only need to draw 1 tree). You **must** use the algorithms described in class for inserting data into a binary search tree.



14. [8 marks] Consider the following binary search tree. Remove the key 15 from the tree and draw the resulting tree. Then remove the key 35 from this new tree and show the final tree (so in the final tree both keys, 15 and 35, have been removed). You **must** use the algorithms described in class for removing data from a binary search tree.