Notes. When you are asked to compute the order of the time complexity function you need to give the **tightest** order. So, 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. Two algorithms, A and B, have time complexities $f_A(n)$ and $f_B(n)$, respectively, where $f_A(n)$ is $O(f_B(n))$ and $f_B(n)$ is not $O(f_A(n))$. Algorithm A is implemented in python, while algorithm B is implemented twice: in java and in C++. The three programs are run on the same computer. Which of the following statements is true?

   (A) The program written in C++ is always faster than the other two programs.
   (B) The program written in python is always slower than the other two programs.
   √(C) There is a value $n_0 \geq 1$ such that the program written in python is faster than the other two programs for every instance of size $n \geq n_0$.
   (D) There is a value $n_0 \geq 1$ such that the program written in java is faster than the other two programs for every instance of size $n \geq n_0$.
   (E) There is a value $n_0 \geq 1$ such that the program written in C++ is faster than the other two programs for every instance of size $n \geq n_0$.

2. 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 false for **every** set of functions $f(n)$, $g(n)$, $h(n)$ as above?
   (A) $f(n)$ is $O(g(n))$
   √(B) $f(n)$ is not $O(h(n))$
   (C) $(f(n) + h(n))$ is $O(g(n))$
   (D) $g(n)$ is not $O(f(n))$
   (E) $(f(n) \times g(n))$ is $O(h(n))$

3. 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** 0
       else {
           $t \leftarrow$`traverse`(left child of r)
           $s \leftarrow$`traverse`(right child of r)
           if $s \geq t$ **then return** $1 + s$
           else return $1 + t$
       }

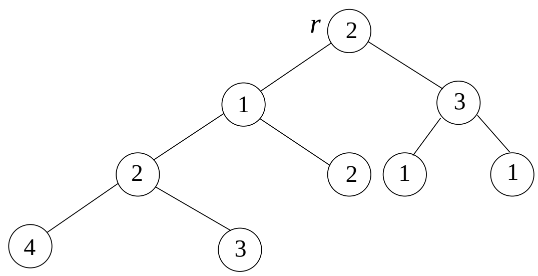
   What does the algorithm do?
   √(A) It computes the height of the tree.
   (B) It computes the number of internal nodes in the largest subtree of T.
   (C) It always returns the value 1.
   (D) It computes the number of nodes in the largest subtree of T.
   (E) It computes the number of internal nodes in the tree.

4. The following algorithm performs a postorder traversal of a proper binary tree and modifies some of the keys stored in the nodes. In the initial call r is the root of the tree.

> **Algorithm** `traverse2`(r)
> **Input:** Root r of a proper binary tree.
>   **if** r is an internal node **then** {
>   `traverse2`(left child of r)
>   `traverse2`(right child of r)
>   **if** (key stored in left child of r) = (key stored in right child of r) **then**
>   increase the key stored in r by 1
>   **else** decrease the key stored in r by 1
>   }

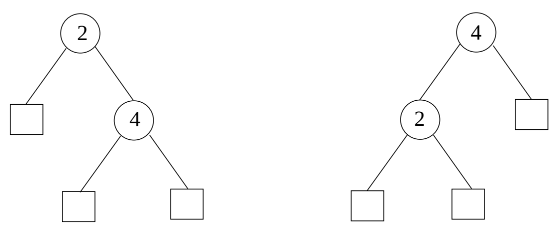
Assume that algorithm `traverse2` is performed over the following tree. After the execution of the algorithm how many nodes will store the key value 1?

(A) 2
(B) 3
√(C) 4
(D) 5
(E) 6



5. The following two proper binary search trees can be built with keys 2 and 4 (the leaves do not store keys). How many different proper binary search trees can be built with the three keys 2, 4, and 6?

(A) 2
(B) 3
(C) 4
√(D) 5
(E) 6



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

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

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

7. 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) $O(i \times n)$
$\surd$(E) $O(n^2)$

8. Consider the following algorithm.

```
Algorithm t(r)
Input: Root r of a tree
   if r is a leaf then return 1
  else {
     tmp ← 0
     for each child u of r do
        tmp ← tmp + t(u)
     return tmp + 1
  }
```

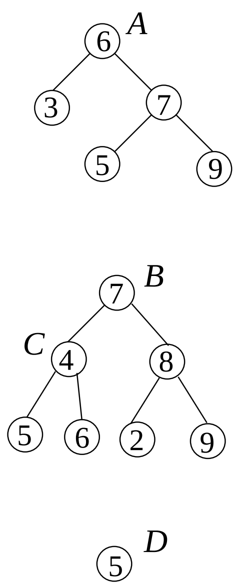
What is the amount of memory needed for the execution stack, if each activation record uses a constant amount c of memory?

(A) The size of the execution stack is at most c, so it uses $O(1)$ space.
(B) The size of the execution stack is at most $c \times \log n$, so it uses $O(\log n)$ space.
(C) The size of the execution stack is at most $c\times$ degree(r), so it uses $O(\text{degree}(r))$ space.
$\surd$(D) The size of the execution stack is at most $c\times$ 1 + height of the tree, so it uses O(height of the tree) space.
(E) The size of the execution stack is at most $c\times$degree(r) $\times n$, so it uses $O(\text{degree}(r) \times n)$ space.

Part 2: Written Answers

9. [15 marks] An internal node u of a proper binary tree is a *search node* if the key stored in its left child is smaller than the key stored in u and the key stored in its right child is larger than the key in u. A proper binary tree is a *partial search tree* if all its internal nodes are search nodes. Leaves also store keys. For example, the trees below with roots A and D are partial search trees, but the tree with root B is not as node C has key 4 and its left child has key 5.

   Write in pseudocode an algorithm `isPartial`(r) that receives as input the root r of a **proper** binary tree and it returns **true** if the tree is a partial search 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() returns **true** if r is a leaf and **false** otherwise.



Algorithm is Partial(r)
In: Root r of a proper binary tree.
Out: True if the tree with root r is a partial serch tree; false otherwise.
if r.isLeaf() **then return** true
else {
    if (r.key $\leq$ r.left.key) **or** (r.key $\geq$ r.right.key) **then**
        return false
     else if isPartial (r.left) $=$ false **then return** false
            else return isPartial (r.right)
}

10. [2 marks] Explain what the worst case for the algorithm is.

    [5.5 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.

    The worst case for the algorithm is when the tree is a partial search tree as then the conditions of the second and third **if** statements will always be false and the algorithm will not terminate early.

    Ignoring recursive calls, the algorithm performs a constant number c_1 of operations when invoked on a leaf and it performs a constant number c_2 of operations when invoked on an internal node. In the worst case the algorithm performs a preorder travesal of the tree, so the algorithm performs one recursive call per node. Therefore, the total number of operations performed by the algorithm is

$$c_1 \times \#\text{leaves} + c_2 \times \#\text{internal nodes} = c_1 \frac{n+1}{2} + c_2 \frac{n-1}{2}.$$

    The time complexity is $O(n)$.

11. [15 marks] Given two arrays A, B each storing n different positive integer values, write in pseudocode an algorithm `union`(A, B, C, n) that stores in a third array C of size $2n$ all the values in A and B without duplicated values. The algorithm must output the number of values that were stored in C. For example, for the following arrays:

A							
8	3	5	1	2	12	11	26

B							
11	9	7	5	8	4	12	1

The algorithm must output the value 11 and C must store the values 8, 3, 5, 1, 2, 12, 11, 26, 9, 7, and 4 (not necessarily in this order). If your `union` algorithm invokes other algorithms, you must write those algorithms also.

Algorithm union(A, B, C, n)
In: Arrays A and B each storing n different positive integers, empty array C of size $2n$.
Out: Number of values stored in array C, which at the end must contain all non-duplicated values from A and B.

// Copy all values of A into C
for $i \leftarrow 0$ **to** $n-1$ **do** $C[i] \leftarrow A[i]$

//Now store in C the values in B that are not in A
$i_C \leftarrow n$
for $i \leftarrow 0$ **to** $n-1$ **do** {
  $i_A \leftarrow 0$
  while $(i_A < n)$ **and** $(A[i_A] \neq B[i])$ **do** $i_A \leftarrow i_A + 1$
  if $i_A = n$ **then** {
    $C[i_C] \leftarrow B[i]$ // $B[i]$ is not in A
    $i_C \leftarrow i_C + 1$
}
return i_C

12. [2 marks] Explain what the worst case for the algorithm is.
[4.5 marks] Compute the worst case time complexity of the above algorithm as a function of m and n. You need to explain how you computed the time complexity.
[0.5 marks] Compute the order ("bigh Oh") of the time complexity.

The worst case for the algorithm is when A and B do not have any common values as then the **while** loop will perform the maximum possible number of iterations.

Each iteration of the first **for** loop performs a constant number c_1 of operations. This loop is repeated n times, so the total number of operations performed by the first loop is $c_1 n$.

Each iteration of the **while** loop performs a constant number c_2 of operations and in the worst case the loop is repeated n times, so the **while** loop performs at most $c_2 n$ operations. Outside the **while** loop but inside the second **for** loop an additional constant number c_3 of operations are performed; hence each iteration of the second **for** loop performs $c_3 + c_2 n$ operations. The second **for** loop iterates n times, hence the total number of operations that it performs is $n(c_3 + c_2 n)$.

Outside the loops a constant number c_4 of operations are performed, thus the total number of opertions performed by the algorithm is $c_4 + c_1 n + n(c_2 n + c_3)$ which is $O(n^2)$.

13. 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:
35, 7, 42, 31, and 21,

[2 marks] (a) when separate chaining is used to resolve collisions

[2 marks] (b) when linear probing is used to resolve collisions

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

Separate chaining

0	→ 35 → 7 → 42 → 21
1	
2	
3	→ 31
4	
5	
6	

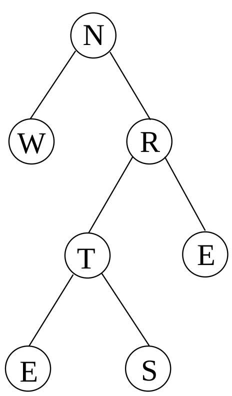
Linear probing

0	35
1	7
2	42
3	31
4	21
5	
6	

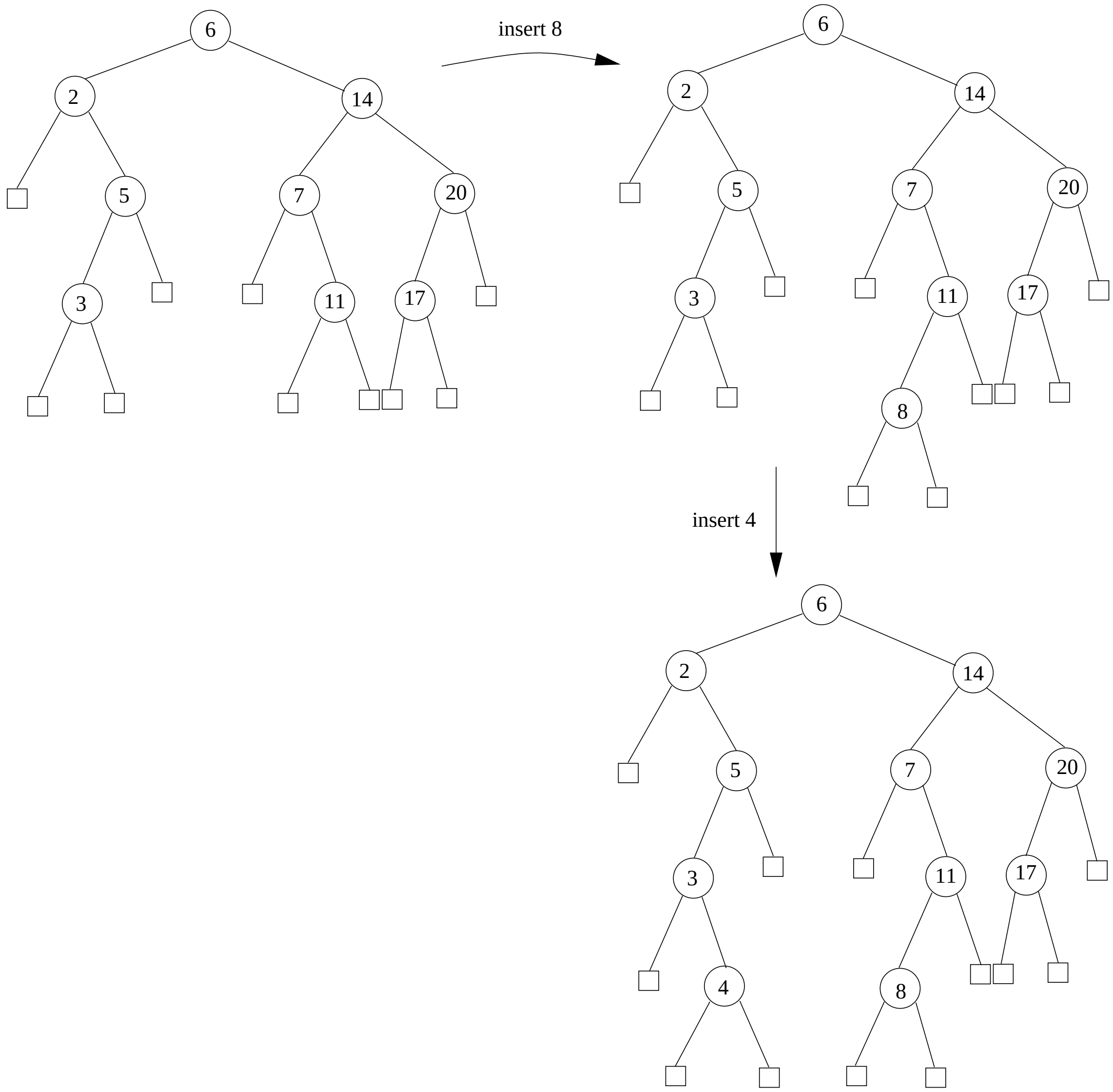
Double hashing

0	35
1	21
2	
3	7
4	31
5	
6	42

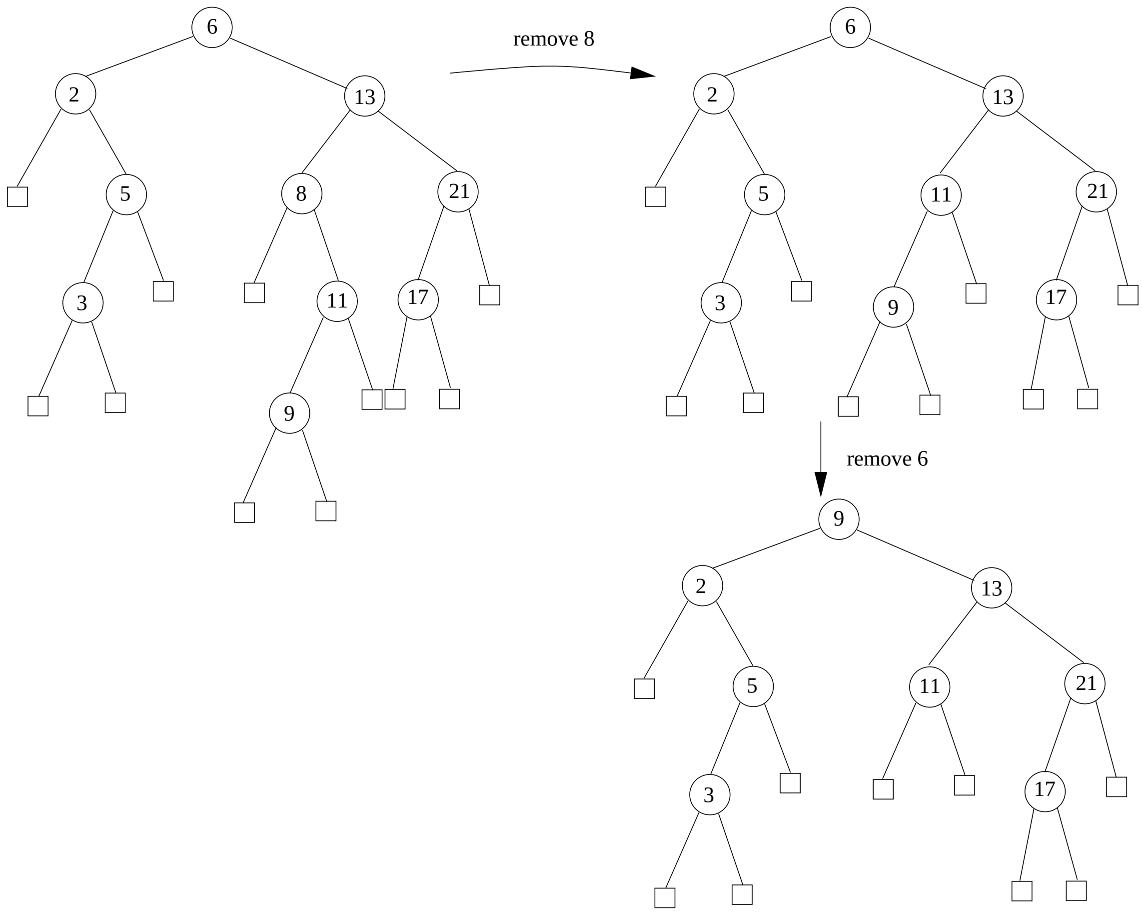
14. [6 marks] Draw a proper binary tree containing the keys E, E, N, R, S, T, and W such that a postorder traversal of the tree visits the nodes in this order: $W\ E\ S\ T\ E\ R\ N$ and a preorder traversal visits the nodes in this order: $N\ W\ R\ T\ E\ S\ E$.



15. [4 marks] Consider the following binary search tree. Insert the key 8 and then insert the key 4 into the tree. Draw the tree after each insertion. You **must** use the algorithms described in class for inserting data in a binary search tree.



16. [5 marks] Consider the following binary search tree. Remove the key 8 from the tree and draw the resulting tree. Then remove the key 6 from this new tree and show the final tree (so in the final tree both keys, 8 and 6, have been removed). You **must** use the algorithms described in class for removing data from a binary search tree.



The University of Western Ontario
Department of Computer Science

PART I	
PART II	
9	
10	
11	
12	
13	
14	
15	
16	
Total	

CS2210A Midterm Examination
November 4, 2017
9 pages, 16 questions
120 minutes

Last Name: ______________________________

Fist Name: ______________________________

CS2210 Class list number: ______________________

Student Number: ___________________________

Instructions

- Write your name, CS2210 class list number, and student number on the space provided.
- Please check that your exam is complete. It should have 9 pages and 16 questions.
- The examination has a total of 100 marks.
- When you are done, call one of the TA's and they will pick your exam up.