Part 1: Multiple Choice

Enter your answers on the Scantron sheet.
We will not mark answers that have been entered on this sheet.
Each multiple choice question is worth 3.5 marks.

Note. 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 $O(n^2)$.

1. Two algorithms, A and B with time complexities $f_A(n)$ and $f_B(n)$, respectively, are to be executed in the same computer. Assume that $f_A(n)$ is $O(f_B(n))$ and $f_B(n)$ is not $O(f_A(n))$. Which of the following statements is true?

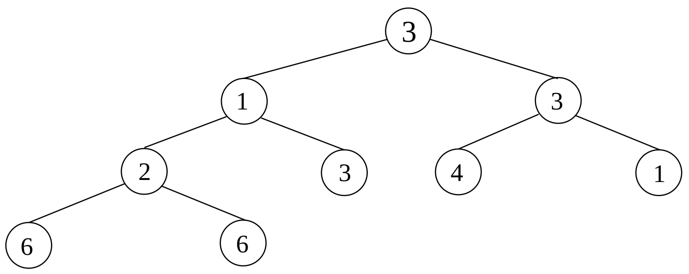
   (A) A is faster than B for all inputs.
   (B) B is faster than A for all inputs.
   $\surd$(C) There is a constant value $n_0 \geq 1$ such that A is faster than B for every input of size $n \geq n_0$.
   (D) There is a constant value $n_0 \geq 1$ such that B is faster than A for every input of size $n \geq n_0$.
   (E) Depending on the computer used, it could be that A is faster than B for all large size inputs, or it could be that B is faster than A for all large size inputs.

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 $h(n) < g(n)$ for all $n \geq 0$. Which of the following statements must be true for **every** triplet of functions $f(n)$, $g(n)$, $h(n)$ as above?
   (A) $f(n)$ is $O(h(n))$
   (B) $g(n)$ is not $O(f(n))$
   (C) $(f(n) + h(n))$ is not $O(g(n))$
   $\surd$(D) $f(n)$ is $O(g(n))$
   (E) $(f(n) \times h(n))$ is not $O(g(n))$

3. Consider the following algorithm.

```
Algorithm eval(r, k)
Input: Root r of a proper binary tree and value k ≥ 0.
    if r is a leaf then return k
    else return eval(left child of r, k + 1) + eval(right child of r, 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
   $\surd$(B) 12
   (C) 9
   (D) 6
   (E) 5



4. Consider the following algorithms, where A is an array storing n positive integer values.

```
Algorithm process($A, n$)
Input: Array $A$ of size $n$
    for $k \leftarrow 0$ to $n-1$ do {
        change($A, n$)
    }

Algorithm change($A, n$)
Input: Array $A$ of size $n$
    $i \leftarrow 0$
    while $i < n$ do {
        $j \leftarrow 0$
        while $j \leq n-1$ do {
            $A[j] \leftarrow A[j] + 1$
            $i \leftarrow i + 1$
            $j \leftarrow j + 1$
        }
    }
```

What is the worst case time complexity of algorithm `process`?
(A) $O(n)$
(B) $O(n \log n)$
[✓ (C) $O(n^2)$]
(D) $O(n^2 \log n)$
(E) $O(n^3)$

5. Let T be a proper binary tree (so each internal node has two children) 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 or $r$)
            $s \leftarrow$traverse(right child or $r$)
            return $2 + s + t$
    }
```

What does the algorithm do?
[✓ (A) It computes the number of edges in the tree.]
(B) It computes twice the number of leaves in the tree.
(C) It computes twice the value of the height of the tree.
(D) It computes twice the total number of nodes in the tree.
(E) It computes the number of internal nodes in the tree plus 2.

6. Let T be a proper binary tree with 7 nodes: a, b, c, d, e, f, g.
A preorder traversal of T visits the nodes in this order: a, b, c, d, e, f, g.
A postorder traversal of T visits the nodes in this order: c, e, f, d, b, g, a.

   Which node is at a distance 3 from the root of T? (**Hint**. In tree T, node d is the right child of node b.)
   (A) a
   (B) b
   (C) c
   (D) d
   $\surd$(E) e

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?
   $\surd$(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. An *Updatable Dictionary* is an ADT that can store a set of n data items with integer keys, and it provides the following four operations: `find(key)`, `smallest()`, `successor(key)` and `increaseKey(key)`. Duplicated keys are not allowed in the dictionary. Operations `find`, `smallest`, and `successor` are as in the Ordered Dictionary ADT that we studied in class.

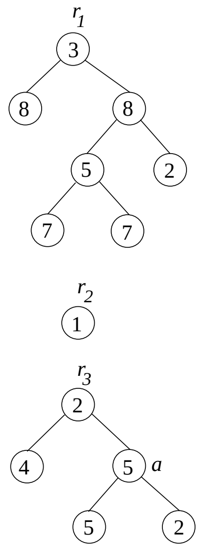
   Operation `increaseKey(key)` finds the data item with the given `key`, if it exists, and it increases its key value by one if doing so does not create duplicated keys; otherwise, no keys are modified. Which of the following statements regarding the most efficient way to implement the above operations of a Updatable Dictionary is true?
   (A) A hash table with a good hash function and separate chaining is the best choice as all operations would have $O(1)$ average running time.
   (B) A binary search tree is the best data structure, as all operations can be performed in $O(\log n)$ time in the worst case, and these trees are much simpler than AVL trees.
   (C) A sorted linked list is the best data structure as `smallest` can be performed in $O(1)$ time in the worst case, while `find`, `successor`, and `increaseKey` need $O(\log n)$ time in the worst case. Furthermore, insertions can be performed in $O(\log n)$ time.
   (D) A hash table using double hashing is the best choice as all operations can be performed in $O(\log n)$ time in average.
   $\surd$(E) A sorted array is the best data structure, as operations `find`, `increaseKey`, and `successor` can be performed in $O(\log n)$ time in the worst case. Operation `smallest` only takes $O(1)$ time in the worst case.

Part 2: Written Answers

Write your answers directly on these sheets. 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 $(n-1)/2$.

9. [14 marks] Consider a proper binary tree where each node p stores a label p.`label`. A proper binary tree is *well labelled* if for every internal node the label of the internal node is different from the labels of its children. For example, the trees below with roots r_1 and r_2 are well labelled, but the tree with root r_3 is not as node a and its left child have the same label, 5.

   Write an algorithm `well_labelled(r)` that receives a input the root of a proper binary tree and returns `true` if the tree is well labelled and it returns `false` otherwise. Use r.`left` and r.`right` to denote the children of r.



```
Algorithm well_labelled($r$)
Input: Root $r$ of a proper binary tree
Output: true if the tree is well balanced; false otherwise

if $r$ is a leaf then return true
else
   if ($r$.label = $r$.left.label) or ($r$.label = $r$.right.label) then
       return false
   else if well_labelled($r$.left) = false then
       return false
   else return well_labelled($r$.right)
```

10. [2 marks] Explain what the worst case for the algorithm is.
    [4.5 marks] Compute the time complexity of the above algorithm in the worst case. 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 input tree is well balanced as then both recursive calls need to be made per node, and hence, the algorithm does not terminate early.

    Ignoring recursive calls, the algorithm performs a constant number c_1 of operations when the node being visited is a leaf, and it performs a constant number c_2 of operations when the node is internal.

    The algorithm performs a preorder traversal of the tree, so the algorithm makes one recursive call per node. Therefore, the total number of operations performed by the algorithm is $c_1\times$ number of leaves + $c_2\times$ number of internal nodes = $c_1\frac{n+1}{2} + c_2\frac{n-1}{2}$. Ignoring constant factors in the time complexity, we see that the complexity is $O(n)$.

11. [14 marks] Given an array A storing m integer values, and an array B storing n integer values, write in pseudocode an algorithm `subarray`(A, B, m, n) that returns the value `true` if A is a sub-array of B and it returns `false` otherwise. A is a sub-array of B iff there is a position $0 \leq j \leq n - m$ such that $A[i] = B[i + j]$ for all $i = 0, 1, \ldots m - 1$. For example, for the arrays A and B shown below the algorithm must return the value `true`, but for the arrays A' and B, the algorithm must return the value `false`.

A	8	4	7

B	1	4	7	5	**8**	**4**	**7**	1	5	5

A'	1	5	8

Algorithm subarray(A, B, m, n)
Input: Array A storing n integer values and array B storing $m \geq n$ integer values.
Output `true` if A is a subarray of B; `false` otherwise.

```
for $i_B \leftarrow 0$ to $n - m$ do {
    $i_A \leftarrow 0$
    while $(i_A < m)$ and $(A[i_A] = B[i_A + i_B])$ do
        $i_A \leftarrow i_A + 1$
    if $i_A = m$ then return true
}
return false
```

12. [2 marks] Explain what the worst case for the algorithm is.
[2.5 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 is not a subarray of B and every value in A, except the last one, is in B. For example, the first $m - 1$ values in A could be all equal to 1 and the last one could be 2, while all values in B are 1. This ensures that the `while` loop performs the maximum number of iterations and the algorithm does not terminate early.

Each iteration of the `while` loop performs a constant number c_1 of operations and the loop is repeated $m - 1$ times in the worst case, as described above, so the number of operations performed by this loop is $c_1(m - 1)$.

Outside the `while` loop, but inside the `for` loop an additional constant number c_2 of operations are performed, so each iteration of the `for` loop performs $c_2 + c_1(m-1)$ operations. Since the `for` loop is repeated $n - m$ times, the total number of operations is $(n - m)(c_2 + c_1(m - 1))$. Ignoring constant terms, the time complexity is $O(nm)$.

13. [7 marks] Consider the following algorithm. A is an array of size n.

```
Algorithm rec(n)
In: Integer value n ≥ 1.
    if $n = 1$ then return $A[0]$
    else {
        $i \leftarrow$ rec($n-1$)
        $A[n-1] \leftarrow A[n-1] + i$
        return $A[n-1]$
    }
```

Write a recurrence equation for the time complexity of this algorithm.

$f(1) = c$, where c is the constant number of operations performed in the base case

$f(n) = c_1 + f(n-1)$ for $n > 1$, where c_1 is the constant number of operations performed by the algorithm ignoring the recursive call.

14. [6.5 marks] Solve the above recurrence equation by repeated substitution. You need to show how you solved the equation.

[0.5 marks] Compute the order of the time complexity.

$$\begin{array}{rcl} f(n) & = & c_1 + f(n-1) \\ f(n-1) & = & c_1 + f(n-2) \\ f(n-2) & = & c_1 + f(n-3) \\ & \vdots & \\ f(2) & = & c_1 + f(1) \end{array}$$

Replacing the value of $f(n-1)$ in the expression for $f(n)$ and then replacing the value of $f(n-2)$ in the resulting expression, and so on, we get

$$f(n) = c_1 + c_1 + c_1 + \cdots + c_1 + f(1) = c_1 + c_1 + c_1 + \cdots + c_1 + c.$$

To determine the number of times that the term c_1 appears in the above summation, note that the above set of equations, for $f(2)$ to $f(n)$, includes $n-1$ equations. Therefore, the term c_1 appears $n-1$ times, and so $f(n) = (n-1)c_1 + c$.

Ignoring constant terms, we get that $f(n)$ is $O(n)$.

15. 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

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

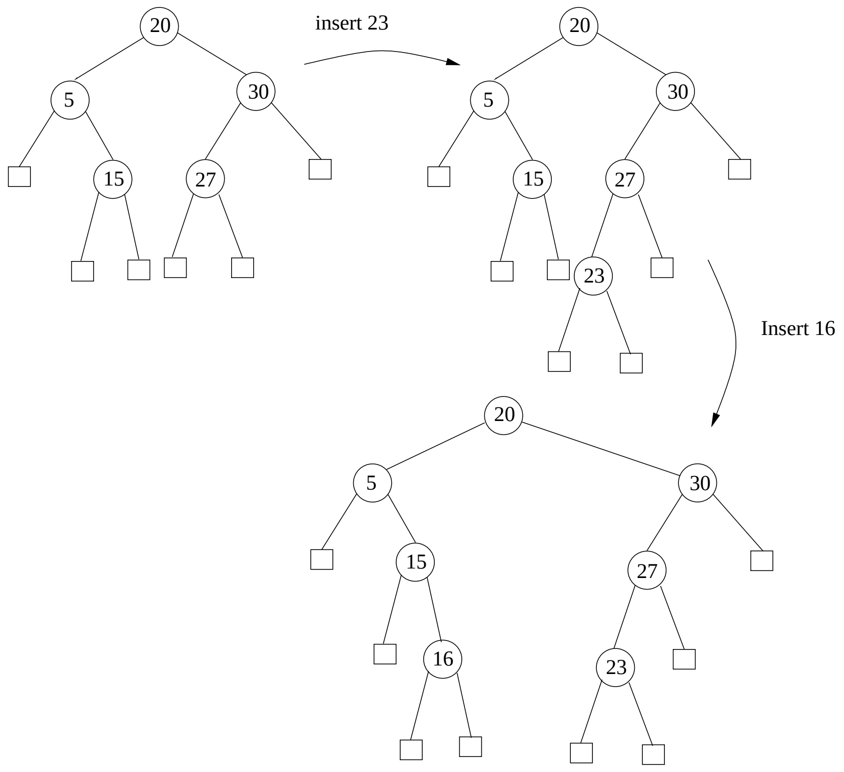
Linear probing

0	62
1	
2	9
3	16
4	2
5	
6	27

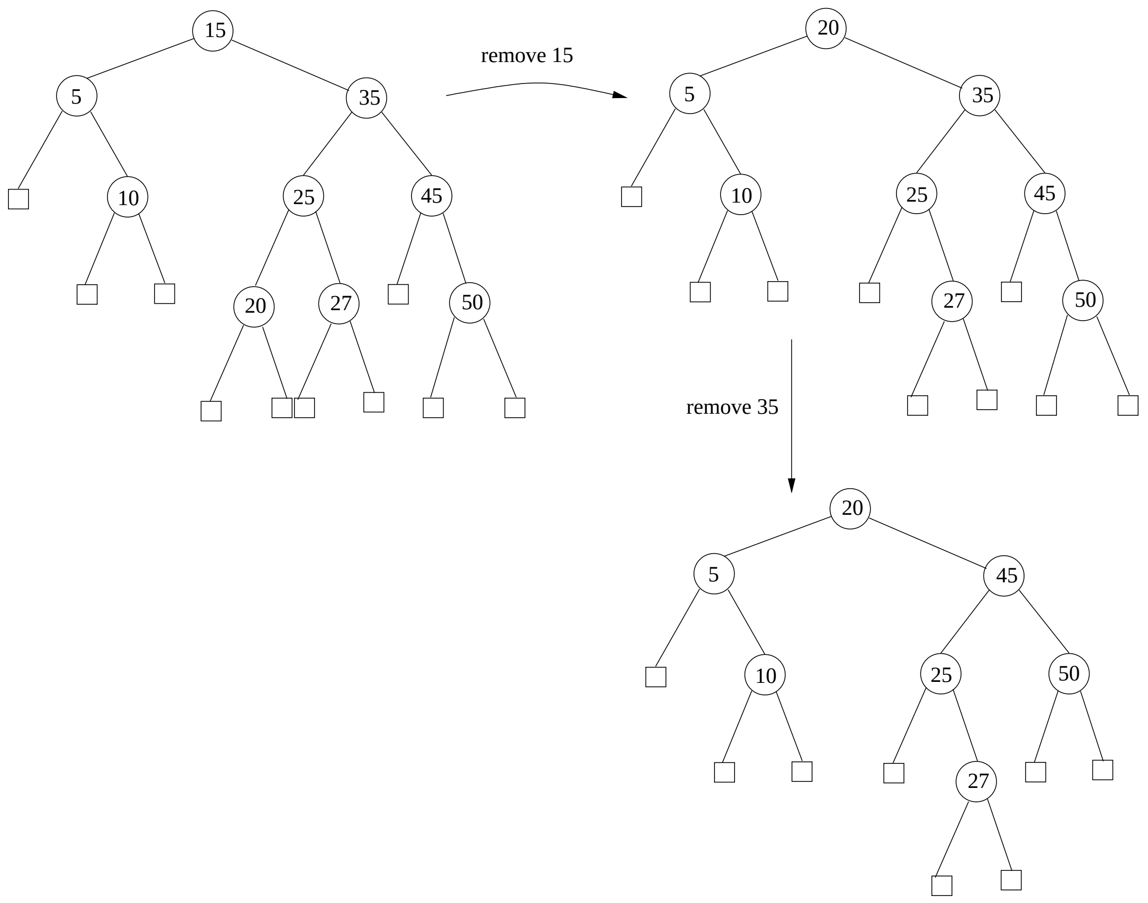
Double hashing

0	
1	62
2	9
3	
4	2
5	27
6	16

16. [2 marks] Consider the following binary search tree. Insert the key 23 and then insert the key 16 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.



17. [4 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.



Western University
Department of Computer Science

PART I	
PART II	
9	
10	
11	
12	
13	
14	
15	
16	
17	
Total	

CS2210A Midterm Examination
November 7, 2016
9 pages, 17 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 17 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.