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.

In all the questions below, $\log(x)$ means $\log_2(x)$. This fact might be useful to you: $\log(x^y) = y\log(x)$.

Note. When you are asked about the time complexity of an algorithm, the best (tightest) characterization of the order of the time complexity is implied. For example, it is correct to say that the time complexity $f_{LS}(n)$ of binary search is $O(n)$. However, even though $f_{LS}(n)$ is also $O(n^2)$, you should not say that the time complexity of linear search is $O(n^2)$.

1. 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 is faster than Q for very large values of n.
   (D) Q is faster than P for very large values of n.
   $\surd$(E) Depending on the implementation and the input, P or Q could be faster.

2. Which of the following functions is **not** $O(n^2)$?
   (A) $1 + \sqrt{n}$
   $\surd$(B) $(n-1)(n\log n - 100)$
   (C) $n^3/(2n^2)$
   (D) $\frac{1}{n^3}$
   (E) $\log(n^3) + 3n^2$

3. What is the worst case time complexity of the following algorithm?

```
Algorithm foo(A, n)
In: Array A storing n integer values
for i ← 0 to n − 1 do {
        j ← 0
        while j < n do {
                if A[j] = i then j ← n − 1
                else {
                        A[j] ← i
                        j ← n − 2
                }
                j ← j + 1
        }
}
```

   (A) $O(i + j)$
   (B) $O(nj)$
   $\surd$(C) $O(n)$
   (D) $O(n^2)$
   (E) $O(n^3)$

4. Consider the order in which the **leaves** of a proper binary tree are visited by preorder, postorder, and inorder traversals. Which of the following statements is true?
(A) In the 3 cases the leaves are visited in different order.
$\surd$(B) In the 3 cases the leaves are visited in the same order.
(C) Only preorder and postorder visit the leaves in the same order.
(D) Only preorder and inorder visit the leaves in the same order.
(E) Only inorder and postorder visit the leaves in the same order.

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

Algorithm `traverse`(r)
In: Root r of a proper binary tree
if r is a leaf **then return** 1
else {
    $t \leftarrow$ `traverse`(left child of r)
    $s \leftarrow$ `traverse`(right child of r)
    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 depth of the nodes.
(D) It computes the height of the tree.
$\surd$(E) It computes the number of leaves in the tree.

6. What is the solution for the following recurrence equation for n even?

$f(n) = 1 + f(n-2)$, for n even and $n > 0$
$f(0) = 1$

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

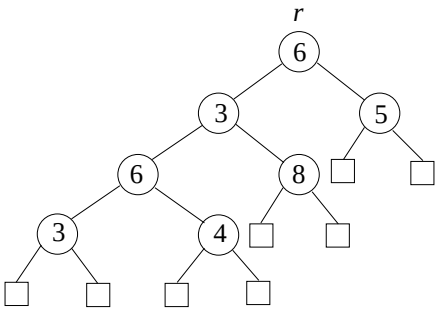
7. A hash table of size n stores n data items. The hash function used by the table maps keys uniformly across the entire table. Which of the following collision resolution strategies minimizes the **worst case** time complexity of the `find` operation?
$\surd$(A) Separate chaining
(B) Open addressing with linear probing
(C) Open addressing with double hashing
(D) Open addressing with either linear probing or double hashing, as both are equally efficient in this case
(E) All collision resolution algorithms give the same worst case time complexity for the `find` operation as we assume a good hash function

8. The following algorithm performs a postorder traversal of a proper binary tree:

Algorithm `change` (r)
In: Root r of a proper binary tree.
if r is an internal node **then** {
    `change` (left child of r)
    `change` (right child of r)
    if (r is not the root) **and** ($r.key < 5$) **then** {
        $p \leftarrow$ parent of r
        $p.key \leftarrow p.key - 2$
    }
}

where $p.key$ is the key stored at node p. Assume that algorithm `change` is performed over the following tree. Square nodes are leaves and they do not store keys. What is the value of the smallest key in the tree after the algorithm finishes?

$\surd$(A) 1
(B) 2
(C) 3
(D) 4
(E) 5

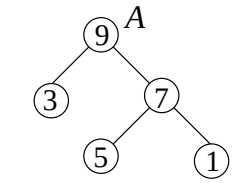


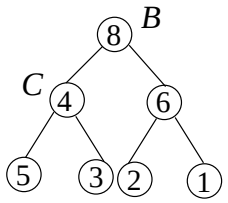
Part 2: Written Answers

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 $(n-1)/2$. $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

9. [14 marks] A proper binary tree is a *heap* if every node stores an integer key and for every internal node the key stored in it is larger than the keys in its children. For example, the trees below with roots A and D are heaps 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 that receives as input the root r of a proper binary tree and it returns **true** if the tree is a heap 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.leaf has value **true** if r is a leaf and **false** otherwise.





Algorithm isHeap(r)
In: Root r of a proper binary tree
Out: True if the tree is a heap and false otherwise.
if r is a leaf **then return true**
else
  if (r.key $\leq$ (r.left).key) **or** (r.key $\leq$ (r.right).key) **then**
    return false
  else
    if isHeap(r.left) = **false then return false**
    else return isHeap(r.right)

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

    [5 marks] Compute the time complexity of the above algorithm in the worst case as a function of the number n of nodes and give its order. Explain how you computed the time complexity.

    The worst case for the algorithm is when the input tree is a heap as then both recursive calls need to be made, and so the maximum number of operations are performed.

    Ignoring recursive calls the algorithm performs a constant number c_1 of operations when called on a leaf and a constant number c_2 when called on an internal node.

    In the worst case the above algorithm performs a preorder traversal of the tree, so the algorithm performs one recursive call per node. Therefore, the total number of operations performed by the algorithm is

$$c_1 \times (\text{n}umber\ of\ leaves) + c_2 \times (\text{n}umber\ of\ internal\ nodes) = c_1\frac{n+1}{2} + c_2\frac{n-1}{2} \text{ is } O(n).$$

11. [14 marks] Given two arrays A and B each storing n **different** integer values we say that B is a *re-arrangement* of A if all values in A are also in B, but the positions of the values in A are different from their positions in B, i.e. for each value $A[i]$ there is a value $B[j]$ such that $i \neq j$ and $A[i] = B[j]$. For example, the array B below is a re-arrangement of A. However, C is not a re-arrangement of A as $A[3] = C[3]$ and neither is D as $A[2]$ is not in D.

Write in pseudocode an algorithm `re-arrangement`(A, B, n) that returns the value `true` if B is a re-arrangement of A and it returns `false` otherwise.

A	7	9	3	1
	0	1	2	3

B	1	7	9	3
	0	1	2	3

C	3	7	9	1
	0	1	2	3

D	9	1	7	2
	0	1	2	3

Algorithm re-arrangement(A, B, n)
In: Arrays A and B storing n different integer values
Out: True is B is a re-arrangement of A and false otherwise
for $i \leftarrow 0$ **to** $n - 1$ **do** {
  $j \leftarrow 0$ // Look for $A[i]$ in B
  while $(j < n)$ **and** $(A[i] \neq B[j])$ **do**
    $j \leftarrow j + 1$
  if $(j = n)$ **or** $(j = i)$ **then return false**
}
return true

12. [2 marks] Explain what the worst case of the algorithm is.

[4 marks] Compute the worst case time complexity of the above algorithm as a function of n and give its order. You need to explain how you computed the time complexity.

The worst case for the algorithm is when B is a re-arrangement of A as in that case the condition of the "if" statement inside the "for" is never true, so the algorithm does not terminate early.

Each iteration of the while loop performs a constant number c of operations. The while loop is repeated at most n times, so the total number of operations performed by this loop is at most cn. Each iteration of the for loop performs a constant number c_1 of operations in addition to the cn operations performed by the while loop. Hence, each iteration of the for loop performs $c_1 + cn$ operations. Outside the for loop a constant number c_2 of operations are performed. Since the for loop is repeated n times, the total number of operations performed by the algorithm is $n(c_1 + cn) + c_2$ is $O(n^2)$.

13. The following algorithm receives as input the root r of a proper binary tree with n nodes. Each node r stores a key, r.key.

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`(left child of r)
    $v_2 \leftarrow$ `proc`(right child of r)
    return r.key$+v_1 + v_2$
}

[2 marks] How many recursive calls does the algorithm perform? Explain.

The algorithm performs a postorder traversal of the tree, so the total number of recursive calls is n.

[2 marks] How much space is needed for each activation record? Explain.

Each activation record needs to store the values of the local variables, parameters and the return address. For the above algorithm each activation record needs to store the values of r, v_1, v_2, and the return address, so each activation record uses constant amount c of space.

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

The maximum number of activation records that are in the execution stack at the same time is equal to the height of the input tree plus 1. This is because every time that a recursive call is made to visit the child of a node a new activation record is stored at the top of the execution stack and every time that a recursive call ends its activation record is removed from the stack. Hence, the space needed by the execution stack is $c\times$ (height of the tree + 1) is O(height of the tree).

14. Consider a hash table of size 7 with hash function $h(k) = k \bmod 7$. Draw the hash table after inserting in it, in the given order, the following values into the table: 10, 17, 6, 20, and 32:

    [2 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.

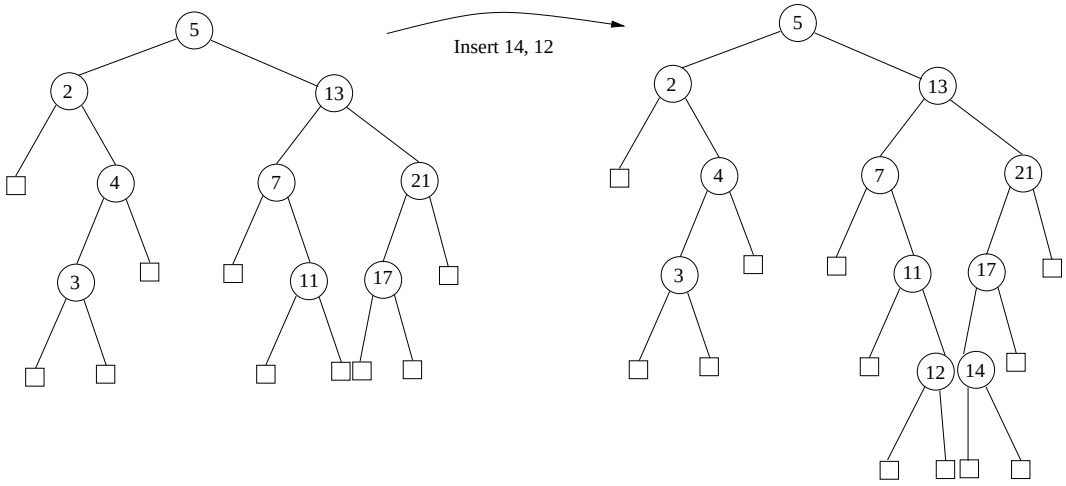
Linear probing

0	20
1	
2	
3	10
4	17
5	32
6	6

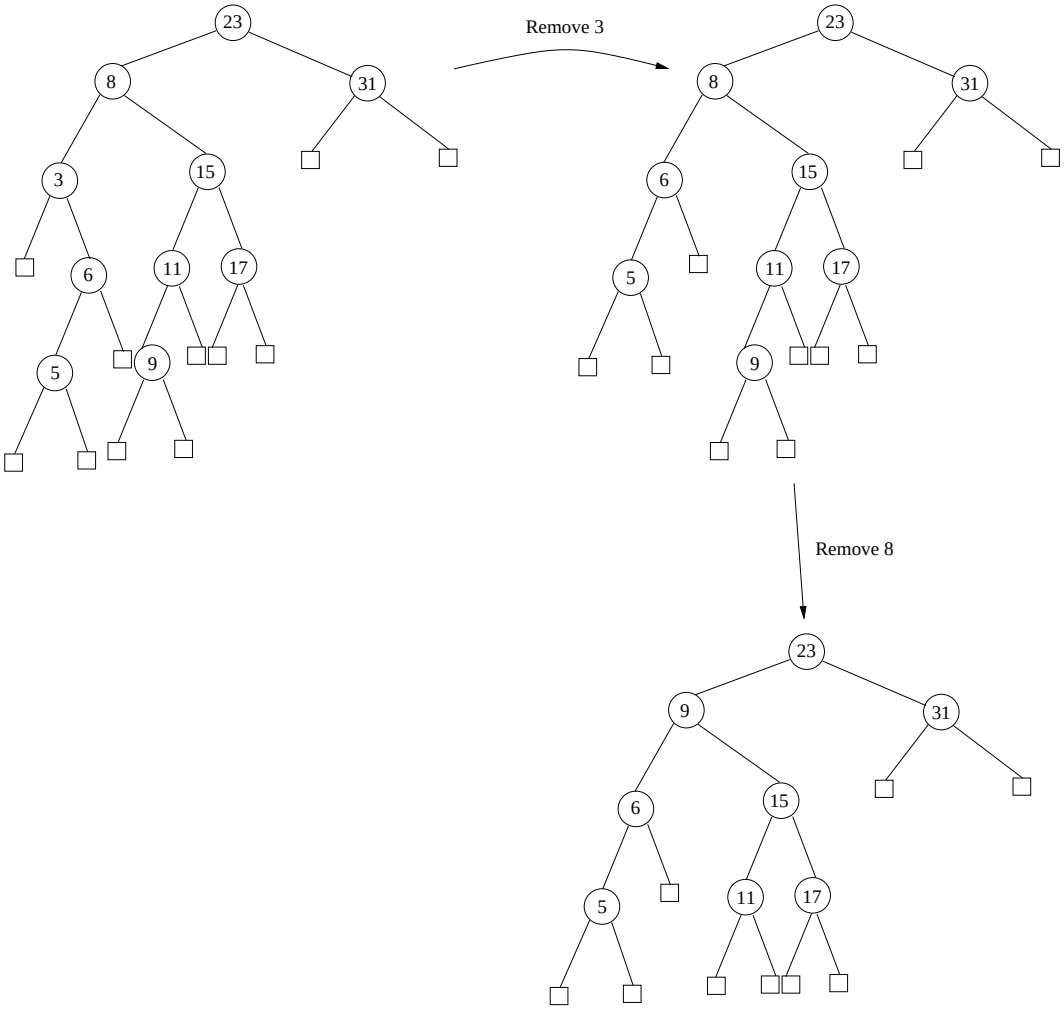
Double hashing

0	6
1	
2	32
3	10
4	20
5	
6	17

15. [3 marks] Consider the following binary search tree. Insert the keys 14 and 12 (in that order) into the tree and show the resulting tree. You **must** use the algorithms described in class for inserting data into a binary search tree.



16. [6 marks] Consider the following binary search tree. Remove the key 3 from the tree and draw the resulting tree. Then remove the key 8 from this new tree and show the final tree (so in the final tree both keys, 3 and 8, 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

CS2210A Midterm Examination
November 7, 2015
9 pages, 16 questions
110 minutes

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

Last Name: _____

First Name: _____

Student Number: _____

Instructions

- Write your name 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 he will pick your exam up.