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: $\log(x^y) = y\log(x)$.

1. Two algorithms, P and Q, have time complexities $p(n)$ and $q(n)$, respectively. If $p(n) = O(q(n))$ and $q(n) = O(p(n))$, then which of the following statements is true?
   (A) When executed on any computer, P and Q will have exactly the same running time (i.e. if both algorithms are started at the same time, they will end at the same moment).
   (B) When executed on any computer, P and Q will have exactly the same running time (i.e. if both algorithms are started at the same time, they will end at the same moment), 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) By just comparing the orders we cannot tell which algorithm is faster. For large inputs, depending on the implementation, P or Q could be faster.

2. Consider a hash table of size M that initially stores n keys. Assuming that the hash function **maps keys uniformly** across the entire table, in the **worst** case how long would it take to insert n additional keys into the table if separate chaining is used to resolve collisions and each entry of the table stores an ordered linked list? Assume that n is much bigger than M.
   (A) $O(n)$ time
   (B) $O(nM)$ time
   (C) $O(n \log n)$ time
   $\surd$(D) $O(n^2/M)$ time
   (E) $O(n^2 \log n)$ time

3. Which of the following functions is **not** $O(n^2)$?
   (A) $3n + 1$
   (B) $(n^2 - 1)/(n + 1)$
   $\surd$ (C) $(3n^2 + 1)(5 \log n - 2)$
   (D) $7/n^3$
   (E) $n \log(n^6)$

4. Let T be a proper binary tree (so each internal node has 2 children) with 7 nodes: a, b, c, d, e, f, g.
   A preorder traversal of T visits the nodes in this order: d, b, c, e, a, g, f.
   An postorder traversal of T visits the nodes in this order: b, a, g, e, f, c, d.

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

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

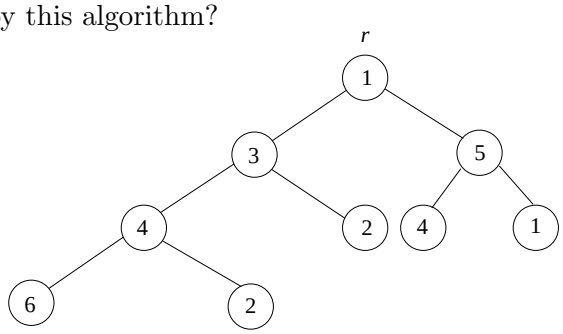
```
Algorithm traverse(r)
Input: Root r of a proper binary tree.
    if r is a leaf then return 0
    else {
          t ←traverse(left child or r)
          s ←traverse(right child or r)
          return 1 + s + t
    }
```

What does the algorithm do?
(A) It computes the height of the tree.
$\surd$(B) It computes the number of internal nodes in the tree.
(C) It computes the total number of nodes in the tree.
(D) It computes the sum of distances from the root to all internal nodes.
(E) It computes the number of leaves in the tree.

6. The following algorithm performs a traversal of a proper binary tree and modifies some of the keys stored in the nodes.

```
Algorithm traverse2(r)
Input: Root r of a proper binary tree.
    if r is a leaf then return the key stored in r
    else {
        a ←traverse2(left child of r)
        if a > key stored in r then Store a as the key of node r
        b ←traverse2(right child of r)
        return a − b
    }
```

Let the algorithm `traverse2` be executed on the following tree. How many nodes will have their keys modified by this algorithm?
(A) 1
(B) 2
$\surd$(C) 3
(D) 4
(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) The resulting binary search tree has the same height, regardless of the order in which the keys are inserted in the tree.
   $\surd$(B) If k_i is the largest key, then in every binary search tree storing the above set of keys, the right child of the node storing k_i is a leaf.
   (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. Consider the following algorithms, where A is an array storing n positive integer values.

```
Algorithm process(A, n)
Input: Array A of size n and value n
    for k ← 0 to n − 1 do
        change(A, n)

Algorithm change(A, n)
Input: Array A of size n and value n
    i ← 0
    while i < n do {
        j ← 0
        while j ≤ n − 1 do {
            A[j] ← A[j] + 1
            i ← i + 1
            j ← j + 1
        }
    }
```

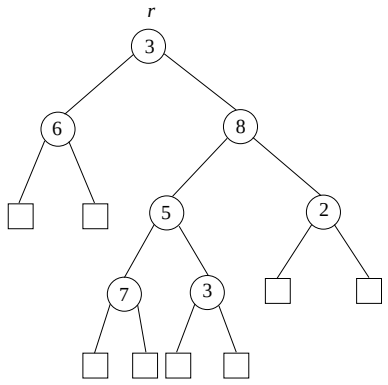
   What is the worst case time complexity of algorithm `process`?
   (A) $O(n)$
   (B) $O(nij)$
   (C) $O(n^2 j)$
   $\surd$(D) $O(n^2)$
   (E) $O(n^3)$

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] Write an algorithm that receives as input an integer value k and the root r of a proper binary tree in which every internal node stores an integer key value, and it outputs the number of nodes storing the key k.

   For example, for the following tree and $k = 3$, the algorithm must output the value 2 as there are two nodes storing the value 3. For the same tree and $k = 4$ the algorithm must return the value 0.



Algorithm find(r, k)
In: Root r of a proper binary tree and integer value k
Out: Number of nodes that store the value k
if r is a leaf **then return** 0
else {
    $n_L \leftarrow$ find (left child of r, k)
    $n_R \leftarrow$ find (right child of r, k)
    if key stored in $r = k$ **then return** $n_L + n_R + 1$
    else return $n_L + n_R$
}

10. [3 marks] Compute the time complexity of the above algorithm in the worst case. Express the complexity as a function of the number n of nodes. You need to give the order of the time complexity.

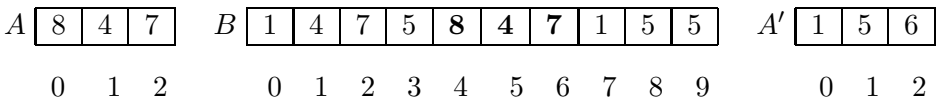
    [3 marks] Explain how you computed the time complexity.

    The above algorithm performs a postorder traversal of the tree, hence the algorithm is called once per each node of the tree.

    Ignoring recursive calls, the algorithm performs a constant number c of operations when invoked on a leaf and it performs a constant number c' of operations when invoked on an internal node. Since the algorithm is invoked once per node, then the total number of operations that it performs is

$$c \times \text{number of leaves} + c' \times \text{number of internal nodes} = c \times \frac{n+1}{2} + c' \times \frac{n-1}{2} \text{ is } O(n).$$

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, n, m) 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` as $A[0] = B[4]$, $A[1] = B[5]$, and $A[2] = B[6]$, but for the arrays A' and B, the algorithm must return the value `false`.



Algorithm subarray (A, B, n, m)

In: Array A storing m integer values, and array B storing n integers; $n \geq m$.

Out: `true` if A is a subarray of B; `false` otherwise

for $i \leftarrow 0$ **to** $n - m$ **do** {

    $j \leftarrow 0$

    while $j < m$ and $A[j] = B[i + j]$ **do**

        $j \leftarrow j + 1$

    if $j = m - 1$ **then return** `true`

}

return `false`

12. [3 marks] Compute the worst case time complexity of the above algorithm. Express the complexity as a function m and n.

    [3 marks] Explain what is the worst case of the algorithm and how you computed the time complexity.

    The worst case for the algorithm is when the first $m - 1$ values in A match with the first $m - 1$ values taken from any subarray of B, but the last value in A is not in B. For example if the first $m - 1$ values in A are 1 and the last value in A is 2, while all the values in B are 1, then the while loop will be repeated $m - 1$ times for each iteration of the for loop.

    In each iteration of the while loop a constant number c of operations are performed and in the worst case the loop is repeated $m - 1$ times, so the total number of operations performed by the while loop is $c(m - 1)$. Beside the while loop, each iteration of the for loop performs an additional constant number c' of operations, so each iteration of the for loop performs $c' + c(m - 1)$ operations. The for loop is repeated $n - m$ times, so the total number of operations performed by the for loop is $(n - m)(c' + c(m - 1))$. Outside the for loop a constant number c'' of additional operations are performed, so the total number of operations performed by the algorithm is

    $(n - m)(c' + c(m - 1)) + c'' = c(m - 1)(n - m) + c'(n - m) + c''$ is $O((n - m)m)$.

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

Algorithm `bal`(n)
In: Integer value n.

```
if n = 0 then return 1
else {
    i ← bal(n − 1)
    A[n] ← i
    return i
}
```

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

Note that when $n = 0$ the above algorithm performs a constant number c' of operations and when $n > 0$ the algorithm also performs a constant number c of operations, if we ignore the recursive calls. Therefore, the time complexity of the algorithm is given by the following recurrence equation:

$f(0) = c'$

$f(n) = c + f(n-1)$, as in the recursive call the value of the parameter is $n - 1$

14. [7 marks] Solve the above recurrence equation by repeated substitution and give the order of the time complexity.

$f(n) = c + f(n-1)$

$f(n-1) = c + f(n-2)$

$f(n-2) = c + f(n-3)$

$\vdots$

$f(1) = c + f(0) = c + c'$

Since the number of equations in the above sequence is n, then the number of times that the term c appears in the sequence is also n. Therefore, $f(n) = cn + c'$ 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

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

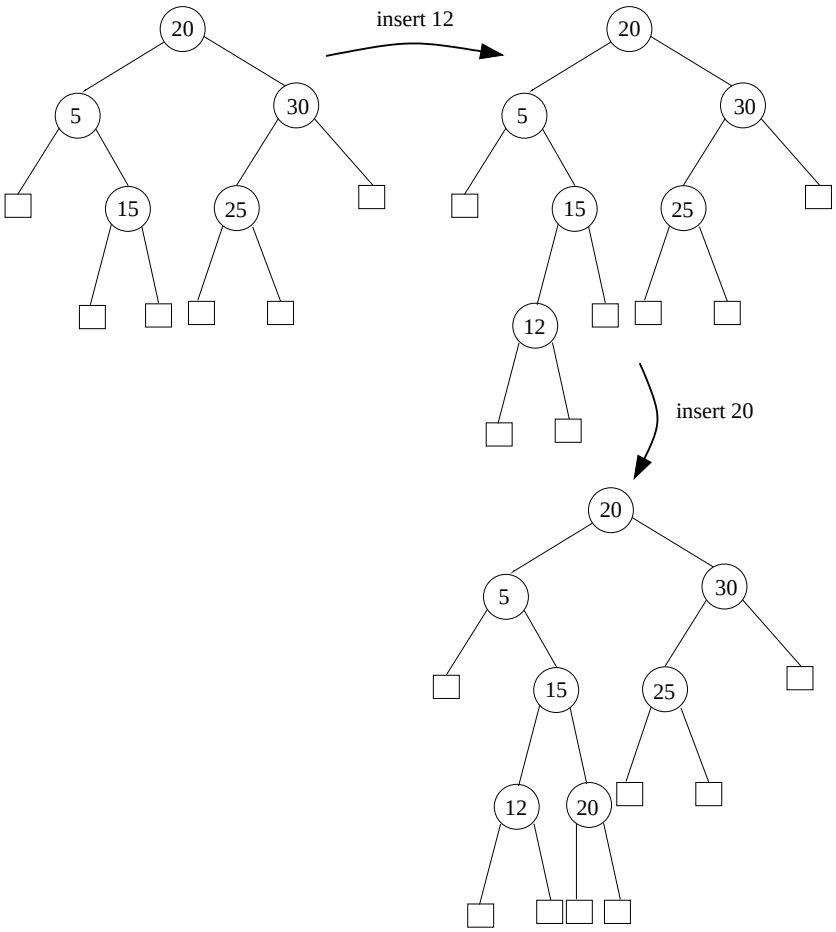
Linear probing

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

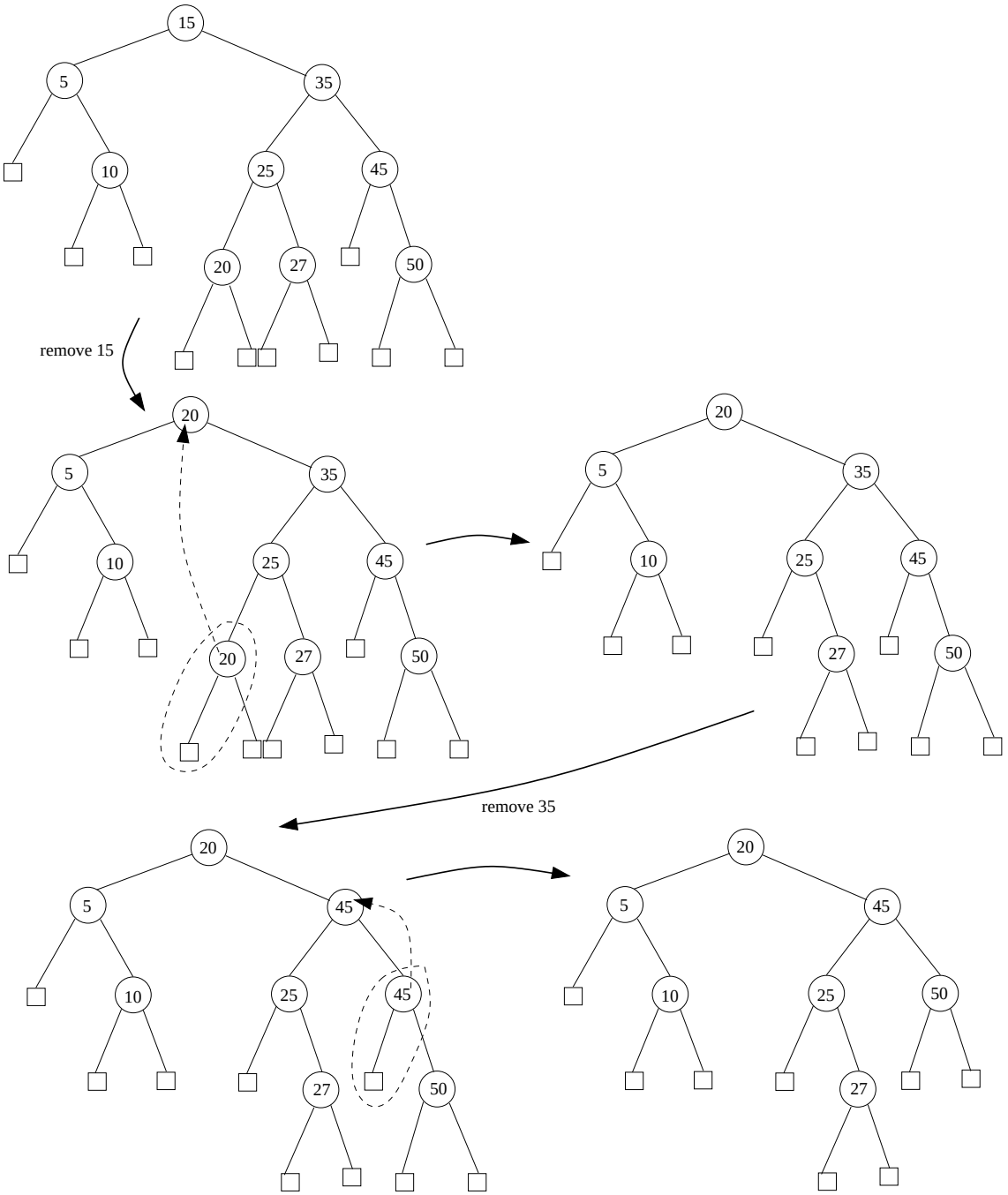
Double hashing

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

16. [3 marks] Consider the following binary search tree. Insert the keys 12 and 20 into the tree. Draw the final tree and **all** intermediate trees that you need. 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.



The University of Western Ontario
Department of Computer Science

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

CS2210A Midterm Examination
October 30, 2012
9 pages, 17 questions
110 minutes

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 17 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.