

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

- Two algorithms, P and Q , have time complexities $p(n)$ and $q(n)$, respectively. Assume that $p(n) = O(q(n))$ and $q(n) \neq O(p(n))$; then, which of the following statements is true?
 - P is faster than Q for every input size n .
 - Q is faster than P for every input size n .
 - There is a value $n_0 \geq 1$ such that Q is faster than P for every input size $n \geq n_0$.
 - There is a value $n_0 \geq 1$ such that P is faster than Q for every input size $n \geq n_0$.
 - Depending on the programming language used to implement P and Q , it could be that P is faster than Q for large size inputs, or it could be that Q is faster than P for large size inputs.
- Which of the following functions is **not** $O(n^3)$?
 - $n^2 + \sqrt{n}$
 - $(n-1)(n \log n + 2)(n+1)$
 - $(n^4 - 1)/(2n^2)$
 - n^{-7}
 - $30n^2 \log(n^7)$
- 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  $r$ )  
    return  $s + t + 1$   
  }
```

What does the algorithm do?

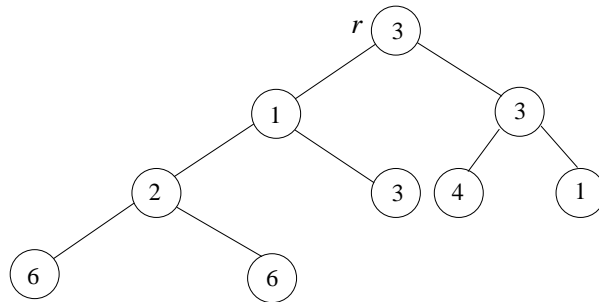
- It computes the height of the tree.
- It computes the number of nodes in the tree.
- It computes the number of nodes in the tree plus 1.
- It computes the number of leaves in the tree.
- It computes the number of internal nodes in the tree.

4. The following algorithm performs an inorder 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)
 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
 `traverse2`(right child of r)
}

Assume that algorithm `traverse2` is performed over the following tree. After the execution of the algorithm how many nodes will store the key value 2?

- (A) 1
(B) 2
 (C) 3
(D) 4
(E) 5



5. A set $K = \{k_1, \dots, k_n\}$ of $n \geq 1$ keys is to be stored in an AVL tree. Which of the following statements is always true?
- (A) If k_i is the smallest key in K , then in every AVL tree storing K the left child of the node storing k_i is a leaf.
- (B) All AVL trees storing K have exactly the same height, regardless of the order in which the keys are inserted in the tree.
- (C) A preorder traversal of any AVL tree storing K visits the keys in increasing order of value.
- (D) In every AVL tree storing K , the key stored at the root of the tree is the same regardless of the order in which the keys were inserted in the tree.
- (E) None of the above statements is always true.
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: b, g, d, a, c, f, e .
An inorder traversal of T visits the nodes in this order: d, g, c, a, f, b, e .
Which node is at a distance 3 from the root of T ? (**Hint.** In tree T , node d is the left child of g .)
- (A) a
 (B) c
(C) d
(E) e
(F) g

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

```
Algorithm PosNeg( $A, n$ )
Input: Array  $A$  of size  $n$ 
  for  $k \leftarrow 0$  to  $n - 1$  do {
    if  $A[k] > 0$  then ProcessPos( $A, n$ )
    else  $A[k] \leftarrow A[k] - 1$ 
  }

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

What is the best characterization of the worst case time complexity of algorithm PosNeg?

- (A) $O(n)$
- (B) $O(nk)$
- ✓(C) $O(n^2)$
- (D) $O(n^2k)$
- (E) $O(n^3)$

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

$$f(1) = 3$$
$$f(n) = f(n - 1) + 1$$

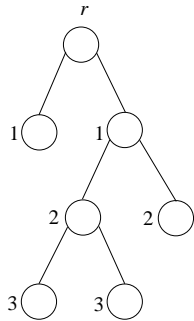
- ✓(A) $f(n) = n + 2$
- (B) $f(n) = n + 3$
- (C) $f(n) = n$
- (D) $f(n) = 2^n + 1$
- (E) $f(n) = 3$

9. An *Updatable Dictionary* is an ADT that can store a set of data items with integer keys, and it provides the following five operations: `insert(key)`, `find(key)`, `smallest()`, `largest()`, and `increaseKey(key)`. Operations `insert`, `find`, `smallest`, and `largest` are as in the Ordered Dictionary ADT that we studied in class. Operation `increaseKey(key)` finds the data item with the given `key` and it increases its key value by one. Which of the following statements regarding the most efficient way to implement an Updatable Dictionary is true? Let n be the number of data items in the Updatable Dictionary.
- (A) A hash table with a good hash function and using 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 array is the best data structure, as operations `insert`, `find`, and `increaseKey` can be performed in $O(\log n)$ time in the worst case. Operations `smallest` and `largest` only take $O(1)$ time in the worst case.
 - ✓(D) An AVL tree is the best data structure, as all above operations can be implemented in $O(\log n)$ time in the worst case.
 - (E) A linked list is the best data structure as `insert` and `increaseKey` can be performed in $O(1)$ time in the worst case, while the other operations only need $O(\log n)$ time in the worst case.

Part 2: Written Answers

Write your answers directly on these sheets.

10. [15 marks] The *total length* of a tree T is the sum of distances from the root of T to all the other nodes of T . For example, the total length of the tree shown below is $1 + 1 + 2 + 2 + 3 + 3 = 12$. In the figure the number beside each node is the distance from the node to the root.



The algorithm has 2 parameters; in the initial call the algorithm is invoked as follows $TL(r, 0)$, where r is the root of the tree whose total distance is going to be computed.

Algorithm $TL(r, d)$

In: Root r of a proper binary tree and integer value d

Out: Total length of the tree

```

if  $r$  is a leaf then return  $d$ 
else {
     $c \leftarrow TL(\text{left child of } r, d + 1)$ 
     $c \leftarrow TL(\text{right child of } r, d + 1) + c$ 
    return  $c + d$ 
}
    
```

11. [3 marks] Compute the time complexity of your algorithm as a function of the number n of nodes in the tree. Give the order of the time complexity.

[2.5 marks] Explain how you computed the time complexity of the algorithm.

First, we ignore the recursive calls. The number of operations that the algorithm performs on a leaf is a constant c ; the number of operations performed on an internal node is also a constant c' . The recursive calls make the algorithm perform a postorder traversal of the tree, hence the algorithm is called exactly once per each node in the tree.

Therefore, the total number of operations performed by the algorithm is

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

12. [15 marks] Let A be an array storing n integer values. Given two values, $0 \leq i \leq j \leq n-1$, the sub-array $A[i : j]$ consists of the entries $A[i], A[i+1], \dots, A[j]$ of A . The *weight*, $w(A[i : j])$, of sub-array $A[i : j]$ is the sum of the values in $A[i : j]$, i.e., $w(A[i : j]) = \sum_{k=i}^j A[k]$. Write an algorithm that given an array A as above it returns the weight of the maximum weight sub-array of A . For example, if A is the array shown below, the sub-array of maximum weight is $A[1 : 3]$ and $w(A[1 : 3]) = 2 - 1 + 4 = 5$. Hence, your algorithm should return the value 5 if given as input this array.

A	-3	2	-1	4	-6	1
	0	1	2	3	4	5

Algorithm maxSubArray(A, n)

In: Array A of size n

Out: Weight of a maximum weight subarray of A

```

max ← A[0]
for i ← 0 to n - 1 do // i: beginning of sub-array
    w ← 0
    for e ← i to n - 1 do // e: end of sub-array
        w ← w + A[e] // w = weight of sub-array A[i..e]
        if w > max then max ← w
    }
return max

```

13. [3 marks] Compute the time complexity of your algorithm and give the order of the time complexity.

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

Every iteration of the inner “for” loop performs a constant number c of operations, and the loop is repeated $n - 1 - i + 1 = n - i$ times, so the total number of operations performed by the second loop is $c(n - i)$. In each iteration of the first loop a constant number c' of operations is performed before the second loop, so each iteration of the first loop performs $c' + c(n - i)$ operations. The first loop is repeated for all values of i from 0 to $n - 1$, so the total number of operations performed by this loop is

$$\sum_{i=0}^{n-1} (c' + c(n - i)) = c'n + c \sum_{i=0}^{n-1} (n - i) = c'n + c \sum_{j=1}^n j = c'n + c \times \frac{n(n+1)}{2}, \text{ is } O(n^2)$$

Outside the first loop a constant number of operations are performed, so the complexity of the algorithm is $O(n^2)$.

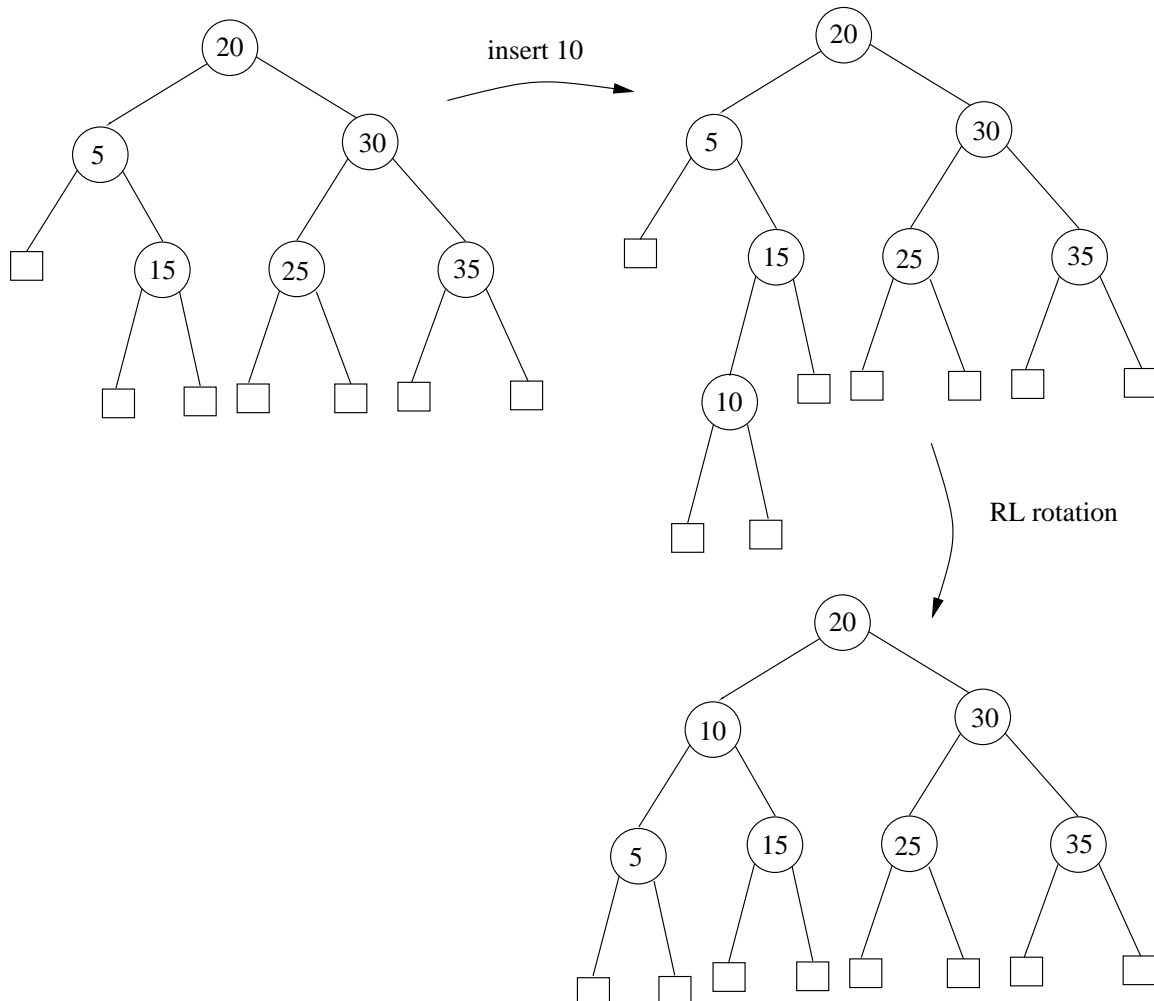
14. Consider a hash table of size 7 with hash function $h(k) = k \bmod 7$. Draw the contents of the table after inserting, in the given order, the following values into the table: 7, 21, 42, 10, and 22:

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

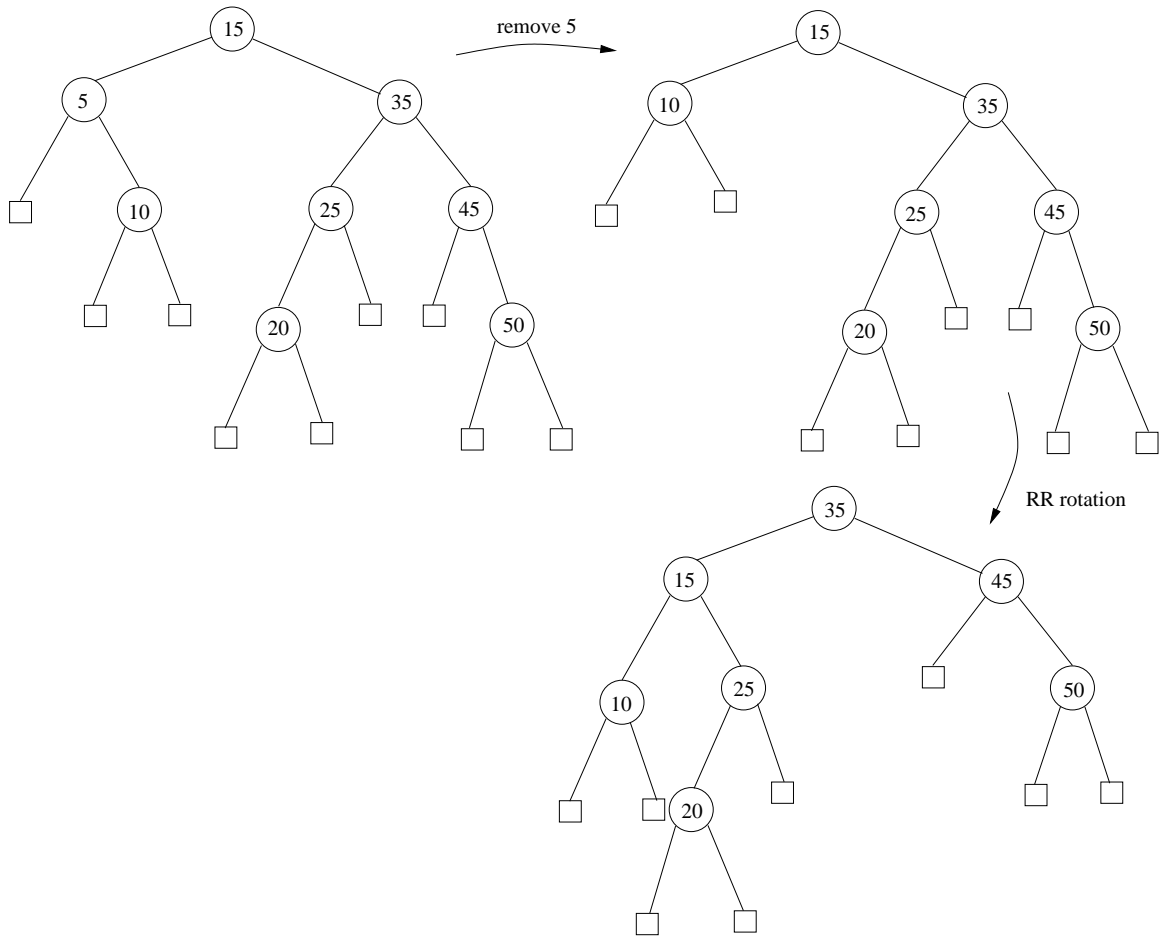
[8 marks] (b) when double hashing with secondary hash function $h'(k) = 5 - (k \bmod 5)$ is used to resolve collisions.

Linear probing		Double hashing	
0	7	0	7
1	21	1	10
2	42	2	
3	10	3	42
4	22	4	21
5		5	
6		6	22

15. [8 marks] Consider the following AVL tree. Insert the key 10 into the tree and rebalance if needed. Draw the final tree and **all** intermediate trees that you need. You **must** use the algorithms described in class for inserting and rebalancing.



16. [8 marks] Consider the following AVL tree. Remove the key 5 from the tree and rebalance if needed. Draw the final tree and **all** intermediate trees that you need. You **must** use the algorithms described in class for removing and rebalancing.



The University of Western Ontario
Department of Computer Science

CS2210A Midterm Examination
November 2, 2010
9 pages, 16 questions
110 minutes

Name: _____

Student Number: _____

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

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 she/he will pick your exam up.