University of Waterloo
Department of Electrical and Computer Engineering
ECE250
Algorithms and Data Structures
Winter 2012

Midterm Examination

Instructor: Ladan Tahvildari, PhD, PEng
Date: Tuesday, February 28, 2012, 5:30 p.m.
Location: RCH 301/307
Duration: 80 minutes
Type: Closed Book

Instructions:

- There are 4 questions. Answer all 4 questions.
- The number in brackets denotes the relative weight of the question (out of 100).
- If information appears to be missing from a question, make a reasonable assumption, state it and proceed.
- Write your answers directly on the sheets.
- If the space to answer a question is not sufficient, use overflow pages.
- When presenting programs, you may use any mixture of pseudocode/C++ constructs as long as the meaning is clear.

Name	Student ID

Question	Mark	Max	Marker
1	A: B: C: D:	40	A: B: C: D:
2	A: B:	20	A: B:
3	A: B: C: D:	30	A: B: C: D:
4	A: B:	10	A: B:
Total		100	

Name: **Student ID:**

Question 1: Algorithm Analysis [40]

Part A [12].

You are given a set of n identical sealed boxes, $n-1$ of which contain one donut and one of which is empty. For convenience, you may assume that $n = 2^k$ for some integer k. To help you find the empty box, you have a balance. At which weighing, you place two sets of boxes on the balance and determine which set is heavier.

i) Describe an algorithm to find the empty box which uses $O(\log n)$ weighings.

a. Divide boxes into two sets of equal size
b. Weigh the sets against each other
c. Report (recursively) with lighter set

ii) Give a recurrence and solve it to show your algorithm runs in $O(\log n)$.

$$W(n) = \begin{cases} 0 & \text{if} \quad n = 1 \\ W(n/2) + 1 & \text{if} \quad n \geq 2 \end{cases}$$

a=1, b=2

$n^{\log_2 1} = n^0 = 1$

f(n)=1=O(1)

Then, W(n)=logn

Name: **Student ID:**

Part B. [10]

We want to extend the asymptotic notations to the case of two parameters *n* and *m* that can go to infinity independently at different rates. For a given function $g(n,m)$, we denote by $O(g(n,m))$ the set of functions:

$O(g(n,m))$= { $f(n,m)$: there exist positive constants c, n_0, m_0
such that $0 \le f(n,m) \le cg(n,m)$ for all $n \ge n_0$ and $m \ge m_0$ }

Give corresponding definitions for $\Omega(g(n,m))$ and $\Theta(g(n,m))$.

$\Omega(g(n, m))$ = {f(n, m): there exist positive constants c, n_0, and m_0
such that $0 \le cg(n, m) \le f(n, m)$
for all $n \ge n_0$ and $m \ge m_0$ }

$\Theta(g(n, m))$ = {f(n, m): there exist positive constants c_1, c_2, n_0, m_0
such that $0 \le c_1g(n, m) \le f(n, m) \le c_2g(n, m)$
for all $n \ge n_0$ and $m \ge m_0$ }

Part C. [8]

Use the Principle of Induction to prove that $5^n - 1$ is divisible by 4. Assume n is a natural number.

Base Case: *To Prove that $5^n - 1$ is divisible by 4, first check it for $n = 1$, $5^1 - 1 = 5 - 1 = 4 = 4 * 1$*

Inductive Step: *Assume it is true for $n = k$ which means $5^k - 1 = 4r$ where r is an integer. Then for $n = k + 1$*

$$5^{k+1} - 1 = 5 * 5^k - 1 = 5 * (4r + 1) - 1 = 4 * 5r + 5 - 1 = 4 * (5r + 1)$$

which, from the Principle of Induction, proves the result.

Name: **Student ID:**

Part D. [10]

Give asymptotic upper and lower bounds for the following recurrence. Assume that $T(n)$ is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answer.

$$T(n) = 4T(n/2) + n^3\sqrt{n}$$

We have $f(n) = n^3\sqrt{n} = n^{7/2}$

Also $n^{\log_b^a} = n^{\log_2^4} = n^2$

Since $n^{\frac{7}{2}} = \Omega(n^{2+\varepsilon}) = \Omega(n^{2+\frac{3}{2}})$ *Case 3 of Master Theorem with* $\varepsilon = 1.5 > 0$

We should also check the Regularity Condition:

$$af(n/b) \le cf(n)$$

$$4(\frac{n}{2})^3\sqrt{\frac{n}{2}} \le cn^3\sqrt{n}$$

$$c = \frac{1}{2\sqrt{2}} \prec 1$$

Case 3 applies, and we have $T(n) = \Theta(n^3\sqrt{n})$

Question 2: Elementary Data Structures [20]

Part A [10].

Give a $\Theta(n)$ – time non-recursive algorithm that reverses a singly linked list of n elements. The algorithm should use no more than constant storage beyond that needed for the list itself.

The constant amount of extra storage will consist of three pointers. Assume that node k is the node currently being visited.

> *"Previous" will be the pointer to the node before k in the linked list (or null if k is the head of the original list).*
> *"Current" will point to k.*
> *"After" will point to the next node of the original linked list (or null if k is the tail).*

Assume that the head of the linked list is h. Assume that all nodes have a member variable called next which points to the next node in the linked list.

```
Previous = null
Current = h
After = null
while ( Current ≠ null )
    After = k.next
    k.next = Previous
    Previous = Current
    Current = After
```

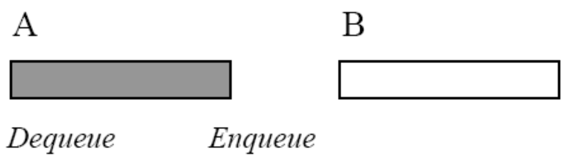
Name: **Student ID:**

Part B. [10]

You have been provided with an implementation of a *queue-of-integers* ADT. This implementation includes the usual operations such as ***Enqueue*** and ***Dequeue***.

Describe how you would implement the Stack ADT using two queues. Specifically, give algorithms for the ***Push*** and ***Pop*** operations. Give tight Big-Oh expressions for the running times of your implementation.

You may assume that ***Enqueue*** and ***Dequeue*** are $O(1)$.



** Elements are currently on A*

pop $\rightarrow$ Simply dequeue of A $\rightarrow O(1)$

push $\rightarrow O(n)$

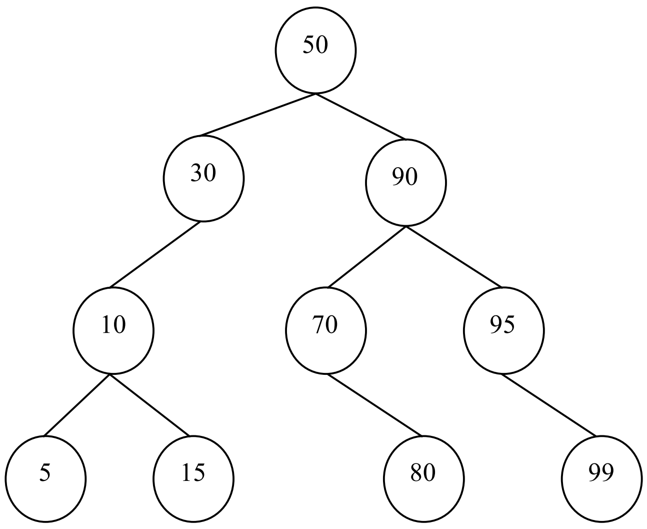
- *Move all elements in A into B $\rightarrow O(n)$*
- *Enqueue the new item; since A is empty, the item will be the first item in A and so LIFO ordered is maintained $\rightarrow O(1)$*
- *Copy back the items in B into A $\rightarrow O(n)$*

Name: **Student ID:**

Question 3: Trees and Tree Traversals [30]

Part A. [7.5]

Consider the binary tree shown below:



For each of the following traversals, list the order in which the nodes are visited.

Pre-order Traversal	*50*	*30*	*10*	*5*	*15*	*90*	*70*	*80*	*95*	*90*
In-order Traversal	*5*	*10*	*15*	*30*	*50*	*70*	*80*	*90*	*95*	*99*
Post-order Traversal	*5*	*15*	*10*	*30*	*80*	*70*	*99*	*95*	*90*	*50*

Part B. [5.5]

Suppose we have the numbers between *1* and *1000* in a BST and we want to search for the number 459. Which of the following sequences could **not** be the sequence of nodes examined?

(1) 902, 278, 404, 663, 410, 540, 421, 459
(2) 40, 279, 905, 837, 421, 422, 813, 459
(3) 1, 491, 481, 431, 450, 475, 474, 456, 459
(4) 984, 165, 884, 203, 885, 207, 599, 459
(5) 767, 352, 761, 361, 758, 390, 457, 459

Give a number from 1 to 5 and *briefly* explain your choice. If no valid explanation is given, no mark will be given.

The correct answer is the fourth (4) sequence.

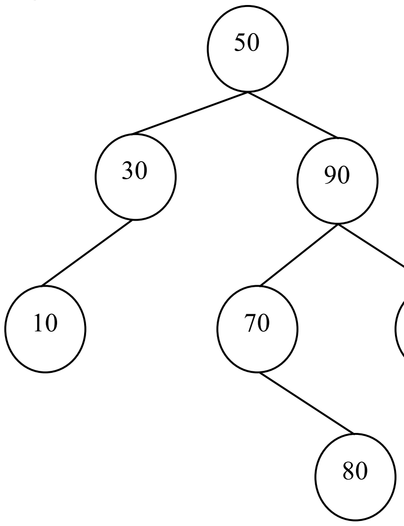
Reason: *Because a BST search always enters a sub-tree of each incorrect node encountered. A legal sequence of examined nodes must have the property that each number in the sequence must be either "less than of all subsequent numbers" or "greater than of all subsequent numbers".*

This is the case with all the sequences above, except for the fourth one, since 884 is less than 885 but greater than 203.

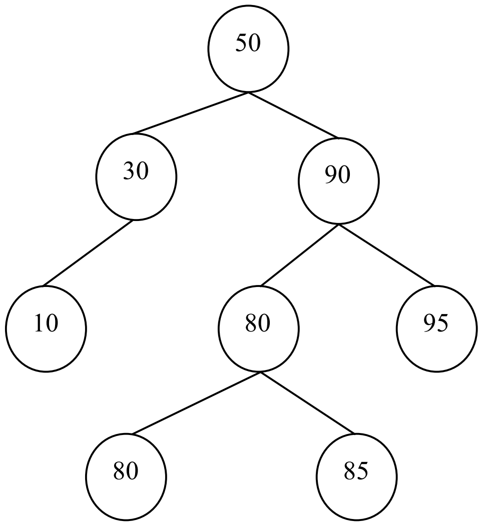
Name: **Student ID:**

Part C. [7]

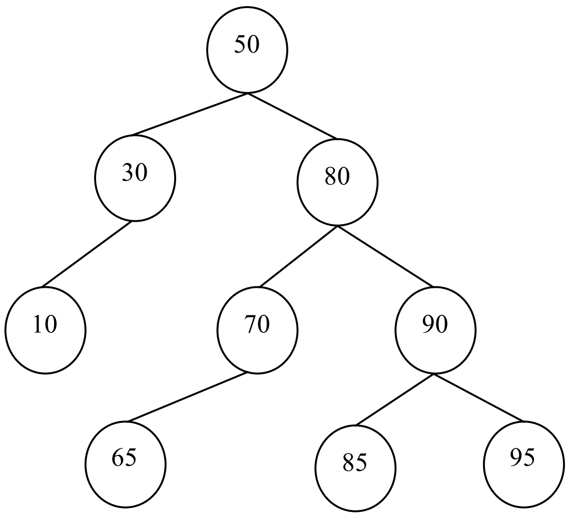
Assume, we have the following AVL tree. Show the tree after inserting 85, 65, and 20.



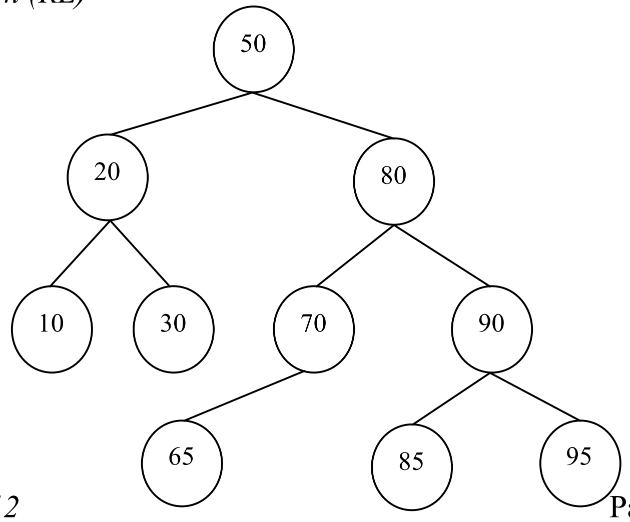
Inserting 85 after a single left rotation (L)



Inserting 65 after a single right rotation (R)



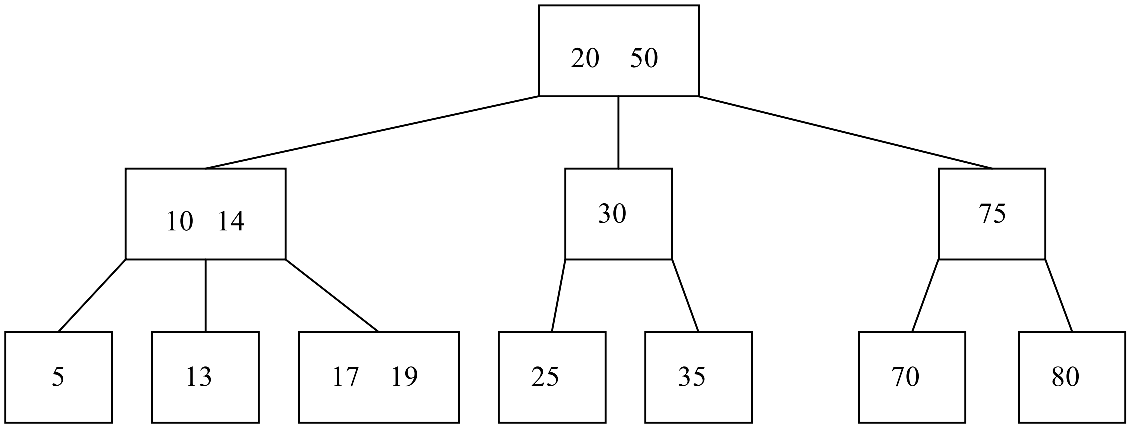
Inserting 20 after a double rotation (RL)



Name: **Student ID:**

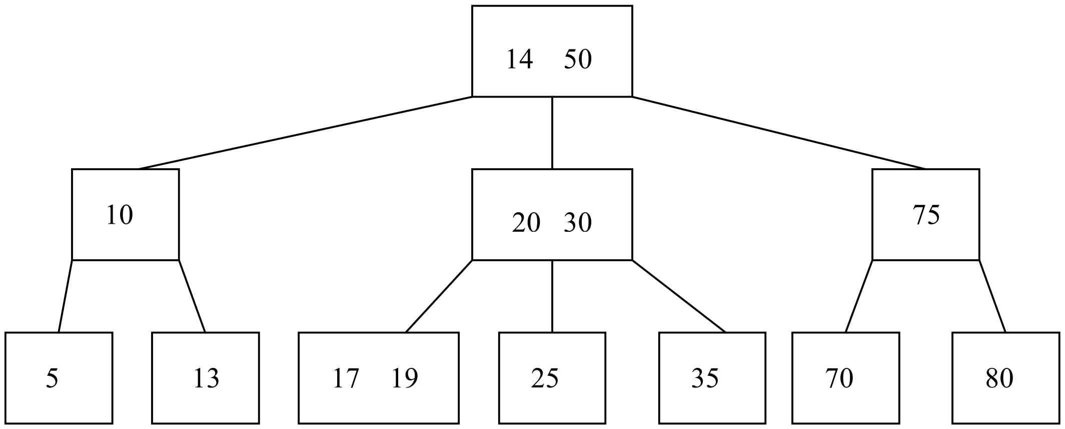
Part D. [10]

Consider the following B-Tree of order 2, called X.

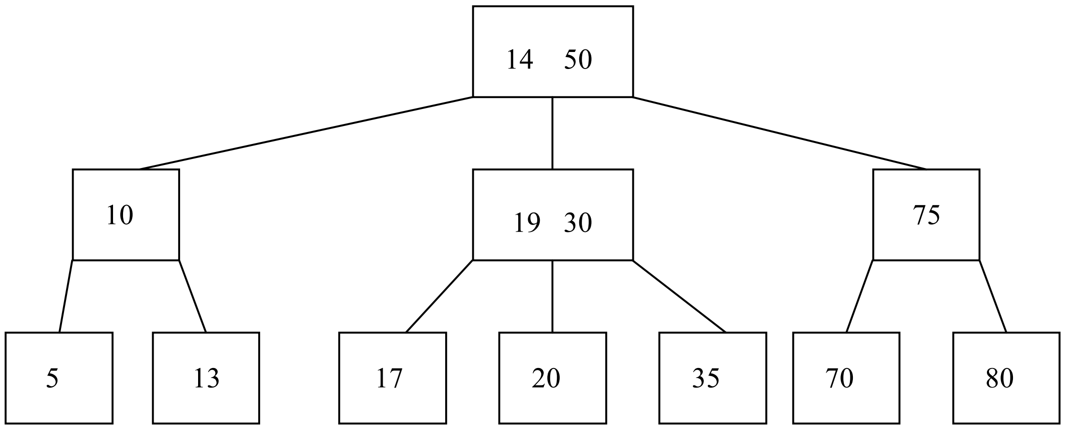


Draw the B-tree that results from deleting 25 from X. Show the tree after each step.

Encounter 30 → Need to transfer



Now to delete 25, we need to transfer



Name: **Student ID:**

Question 4: Hashing [10]

Consider the following hash table of size 13.

0	13
1	
2	2
3	
4	30
5	
6	
7	20
8	
9	139
10	10
11	11
12	25

For the following questions, indicate the insertion positions under the specified hashing scheme. You can assume the (primary) hash function is $h(k) = k \mod 13$.

Part A - Linear Probing [3]

61 would be inserted at position ___*1*___.

61 mod 13 = 9 → full; the first available slot is 1.

and if 40 was inserted after, it would be placed at position ___*3*___.

40 mod 13=1 → full; the first available slot is 3.

Part B - Double Hashing [7]

Using the second hash function $h_2(k) = ((k+2) \mod 5) + 1$

61 would be inserted at position ___*8*___.

$h_2(61) = ((61+2) \mod 5) + 1 = 3 + 1 = 4$

61 mod 13 = 9 → full.

(9 + 1x4) mod 13 = 0 → full

(9 + 2x4) mod 13 = 4 → full

(9 + 3x40 mod 13 = 8

and if 40 was inserted after, it would be placed at position ___*1*___.

40 mod 13=1

Name: **Student ID:**

OVERFLOW SHEET [Identify the question(s) being answered.]

Name: **Student ID:**

OVERFLOW SHEET [Identify the question(s) being answered.]