ECE368 Exam 1
Spring 2016

Thursday, March 10, 2016
15:00-16:15pm
ARMS 1010

READ THIS BEFORE YOU BEGIN

This is a *closed-book, closed-notes* exam. Electronic devices are not allowed. The time allotted for this exam is exactly 75 minutes.

Always show as much of your work as practical - partial credit is largely a function of the *clarity and quality* of the work shown. *Be concise*. It is fine to use the blank page opposite each equation (or at the back of each question) for your work. Do draw an arrow to indicate that if you do so.

This exam consists of 9 pages; please check to make sure that all of these pages are present before you begin. Credit will not be awarded for pages that are missing – it is *your responsibility* to make sure that you have a complete copy of the exam.

IMPORTANT: Write your login at the TOP of EACH page. Also, be sure to *read* and *sign* the *Academic Honesty Statement* that follows:

"In signing this statement, I hereby certify that the work on this exam is my own and that I have not copied the work of any other student while completing it. I understand that, if I fail to honor this agreement, I will receive a score of ZERO for this exam and will be subject to possible disciplinary action."

Printed Name:

Login:

Signature:

DO NOT BEGIN UNTIL INSTRUCTED TO DO SO …

Each line receives:
1 point if perfect.
½ point if close but not precise.
0 points if completely off.
Error does not carry over next line.

1. **Analysis of algorithms (20 points total)**

Consider the following procedure that performs multiplication of two upper triangular matrices $A[1 \ldots n][1 \ldots n]$ and $B[1 \ldots n][1 \ldots n]$.

MATRIX_MULTIPLY ($A[1 \ldots n][1 \ldots n], B[1 \ldots n][1 \ldots n]$)	*Cost*	*Times*
for ($i = 1;\ i \le n;\ i{+}{+}$) {	C_1	$n + 1$
for ($j = 1;\ j \le n;\ j{+}{+}$) {	C_2	$n*(n + 1)$
$c_{ij} \leftarrow 0$	C_3	$n * n$
for ($k = i; k \le j; k{+}{+}$) {	C_4	$\sum_{i=1}^{i=n} \sum_{j=1}^{j=n} j - i + 2$
$c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$	C_5	$\sum_{i=1}^{i=n} \sum_{j=1}^{j=n} j - i + 1$
}		
}		
}		
return C	C_6	1

a) **(6 points)** For each instruction, fill the "Times" column. Write down the expression for the number of times the instruction is executed in terms of i, j, k, and n (you may not need all of these terms).

b) **(4 points)** What is the worst case asymptotic time complexity of the algorithm?

Other polynomial receives 2 points.
Log(n) receives 1 point.
No answer 0 points.

$O(n^3)$

FUNCTION_B (int n)	*Cost*	*Times*
int sum = 0;	C_1	1
for ($i = 1;\ i \le n;\ i \mathrel{*}= 2$)	C_2	$\log n + 2$
for ($j = 0;\ j < i;\ j{+}{+}$)	C_3	$\sum_{i=0}^{i=logn} (2^i + 1)$
sum++;	C_4	$\sum_{i=0}^{i=logn} 2^i$
return sum	C_5	1

Explanation in the next page

c) **(5 points)** For each instruction, fill the "Times" column. Write down the expression for the number of times the instruction is executed in terms of i, j and n (you may not need all of these terms).

d) **(5points)** What is the worst case asymptotic time complexity of FUNCTION_B?

$O(n)$

Right answer: 5 pts.
Other polynomials or logs: 4 pts.
No answer: 1 pts.

a) $T_4 = \sum_{i=1}^{n}\sum_{j=1}^{n}(j-i+2) = \sum_{i=1}^{n}\left(\frac{n(n+1)}{2} - i + 2\right)$

$= \frac{n^2(n+1)}{2} - \frac{n(n+1)}{2} + 2n = \frac{n^3}{2} + \frac{3n}{2}$

$T_5 = \sum_{i=1}^{n}\sum_{j=1}^{n}(j-i+1) = \frac{n^2(n+1)}{2} - \frac{n(n+1)}{2} + n = \frac{n^3}{2} + \frac{n}{2}$

b) $O(n^3)$

c) T_2 : $i = 1, 2, 4, \cdots, n$

$= 2^0, 2^1, 2^2, \cdots, 2^{\log_2 n}$

$\therefore$ # of $i = \log_2 n + 1$

$T_2 = \log_2 n + 2$

T_3 : for each i

j runs 2^i

$\therefore$ # of iters $= \sum_{i=0}^{\log_2 n} 2^i$

$T_3 = \sum_{i=0}^{\log_2 n}(2^i + 1)$

$= \frac{2^{(\log_2 n + 1)} - 1}{2-1} + (\log_2 n + 1)$

$= 2^{\log_2 n} \cdot 2 + \log_2 n = 2n + \log_2 n$

$T_4 = \sum_{i=1}^{\log_2 n}(2^i) = 2^{\log_2 n} \cdot 2 = 2n$

d) $T = T_1 + T_2 + T_3 + T_4 + T_5$

$= O(n)$

2. Class Participation Type Algorithms (20 points total)

a) **(10 points)** Explain how you will generate a uniformly distributed random integer between 1 and 1000 given only one coin. You can flip the coin multiple times and assume the coin is a fair coin.

Full credit (10 pts):

Make each coin flip be a binary number. Flip coin 10 times, to create a 10-bit digit number (or traverse a complete b-tree). Repeat process if n>1000.

If forgot to handle n>1000. (9 pts)

Partial credit for most common answers (7 Pts)

-Binary Search Approach: Start at 500. If coin is H, look at ranges [1,500]. If tails, look at [501,1000]. Keep dividing in 2 and flipping coin until convergence. <u>(Middle numbers are biased/unbiased –depending on implementation)</u>

-Make coin be a binary number. Flip coin 1000 times and add the value of each flip (0 or 1). <u>(Small numbers are biased)</u>

Partial credit will vary depending on robustness.

No answer = 0.

b) **(10 points)** Given the array J = [1,2,3,4,5,6,7,8,9,10], explain in what order you would add the elements in J to create a binary search tree with minimal height.

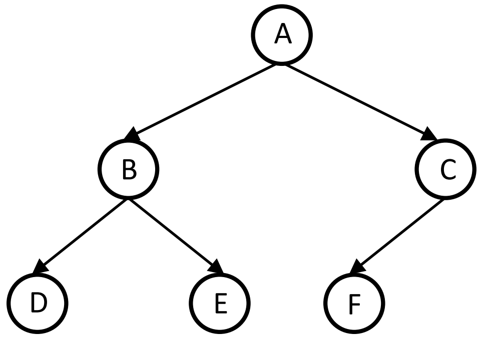
Insert Median First, then split in half, and insert median (recursively) = 10 points.

Draws a minimal height binary tree = 10 points.

Wordy and inaccurate answer = 5 or below.

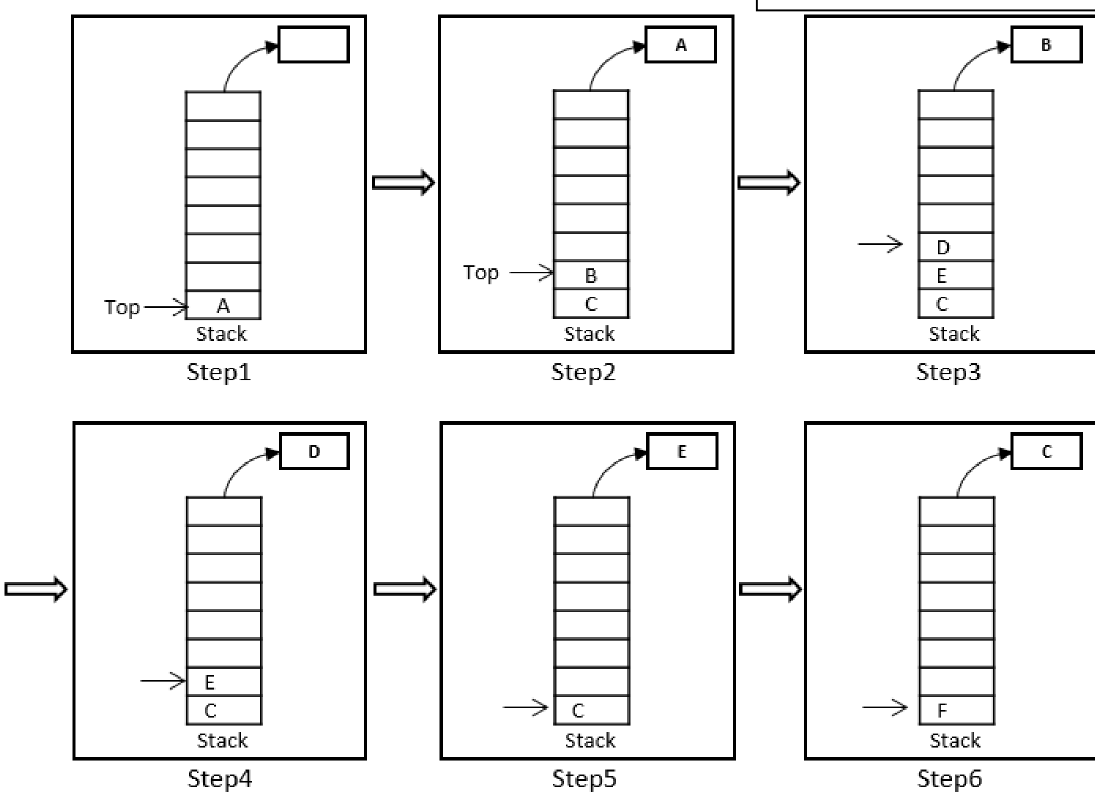
Iterative Tree Traversal Using Stacks and Queues (20 points total)

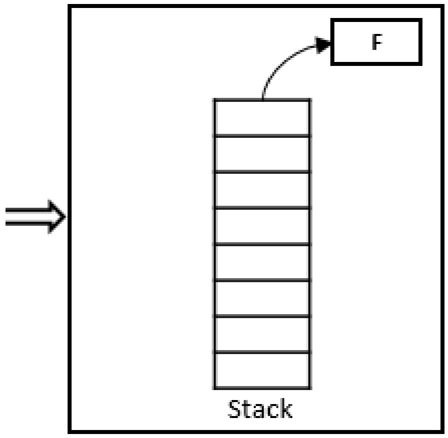
Consider the following binary tree:



a) **(10 points)** Consider a **preorder** traversal of the given binary tree using a stack. Whenever an element is popped up from the stack, the element will be printed out and its children will be pushed to the stack (if not NULL). Fill in the stack elements, update stack top, and write the popped elements box in Steps

Each "printed" letter = 1 point if correct.
Each stack = 1 point if correct.
If didn't update "Top" = -2 points.

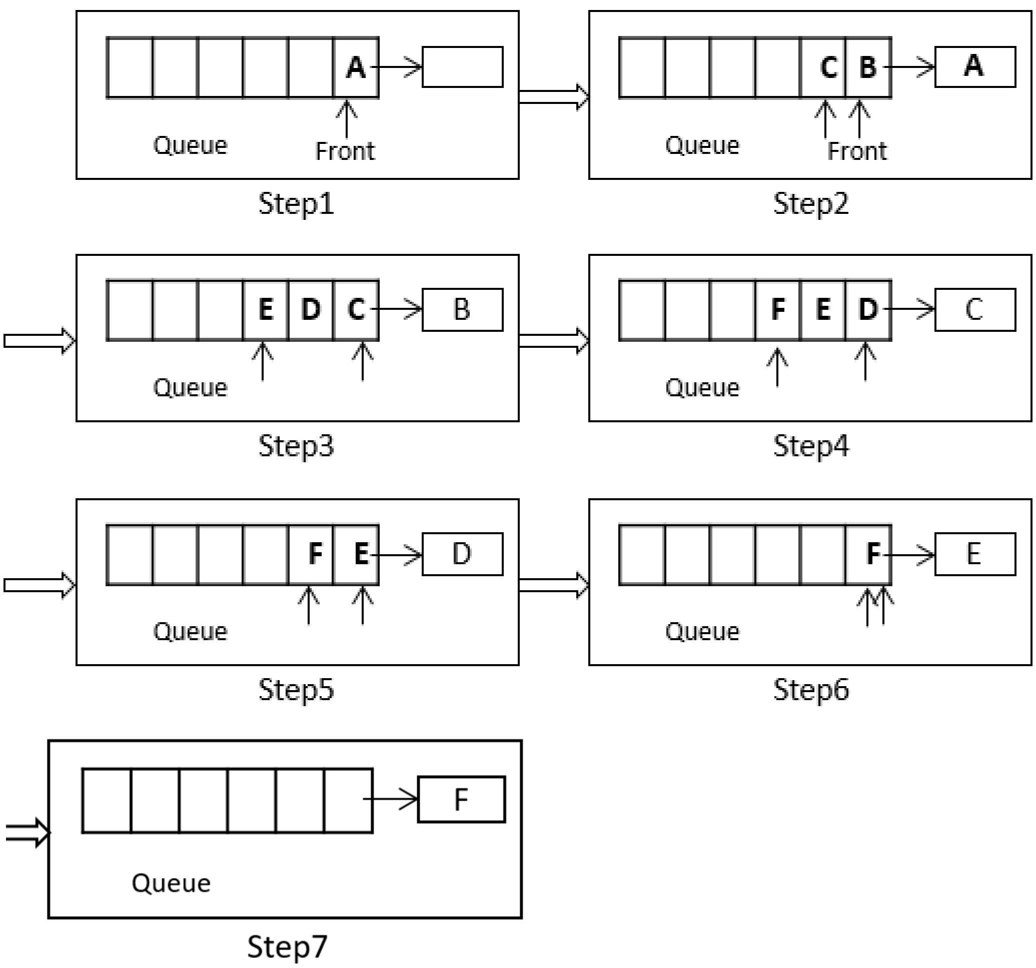




Step7

b) **(10 points)** Consider using a queue for **<u>Breadth-First Search</u>** (BFS) traversal of the given binary tree in the previous question. Whenever an element is dequeued from the queue, the element will be printed out and its chil
in the queue elements, update front & back, and v
3-7.

Each printed letter = 1 point if correct.
Each queue = 1 point if correct.
If didn't update "F & B" = -2 points.



No answer = 1
Yes, = 3
No, but not a good reason = 5-8
No, and the reason similar to the solution = 10.

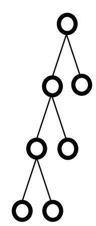
3. **Perfect Binary Trees (20 points total)**

Consider the following condition C1:

C1: Every node is either a leaf or a full node.

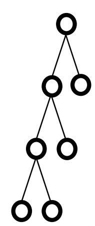
a) **(10 points)** Is every tree satisfying C1 a perfect tree? Justify your answer.

No,
- The possible reasons
  - An unbalanced tree may satisfy C1 but is not a perfect tree.
  - The tree satisfying C1 may not satisfy the property of the perfect tree:
    - A perfect tree has even leaf nodes
    - All leaf nodes of a perfect tree have the same depth.
  - Counter example:



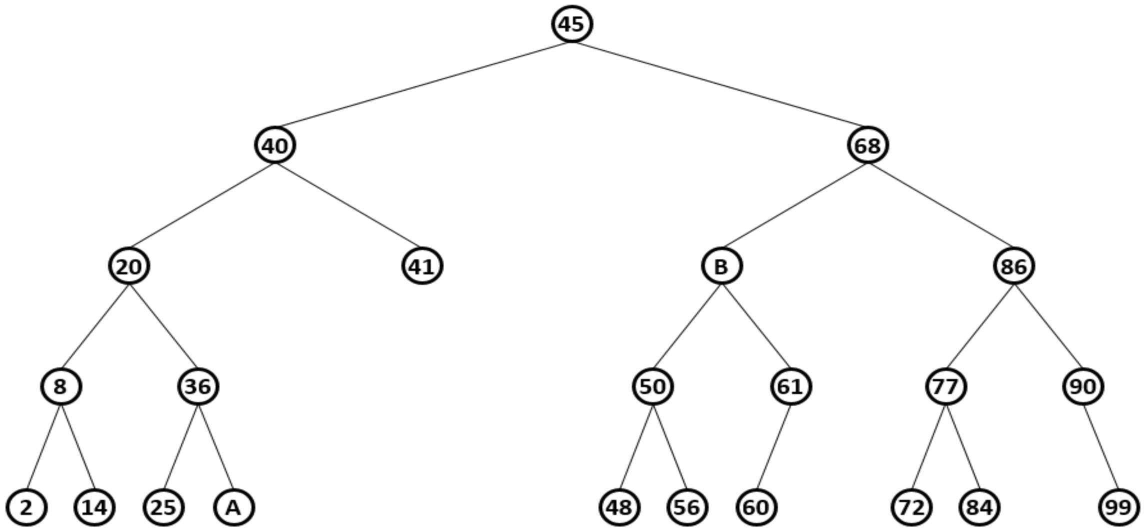
b) **(10 points)** Does every binary tree satisfying C1 have height $h = \theta(\lg n)$? Justify your answer.

No,
- The possible reasons
  - An unbalanced tree may satisfy C1 but does not have height $h = \theta(\lg n)$. The height may be $\theta(n)$.
  - Counter example:



4. **Binary Search Tree (20 points total)**

Consider the following binary search tree.



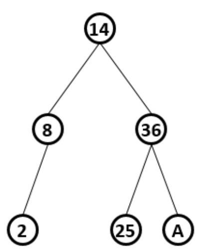
a) **(10 points)** Write all integer values possible for A and B (we assume the tree does not have duplicate values).

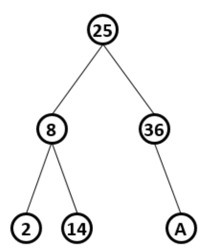
A: 37, 38, 39

B: 57, 58, 59

For each part, No answer = 0
Answer which does not include the numbers = 1
Answer which includes the numbers but is not correct = 2
Correct Answer = 5

b) **(10 points)** Draw the final binary search tree after the following operation: delete 20 (you do not have to modify A and B with numbers). Use either **inorder successor** or **inorder predecessor** to perform node deletion.





No answer = 0
Answer which does not use either inorder successor or presecessor = 5
Answer which uses inorder successor or presecessor but has the other shape = 8
Correct Answer = 10

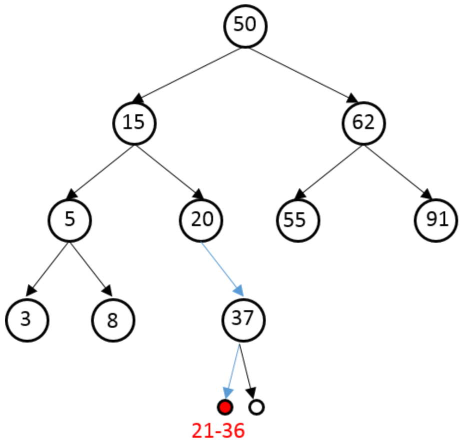
c) (**Bonus Question – 10 points**) Write C or C++ code to implement a function that takes as input a pointer to a binary sear passed by reference. The functio position can take. If the bool is otherwise the right child position

Correct code = 10 points
Correct approach, but minor mistakes in code = 6-8 points
Similar approach, but minor mistakes in code = 3-5 points
Others / no code = 0 point

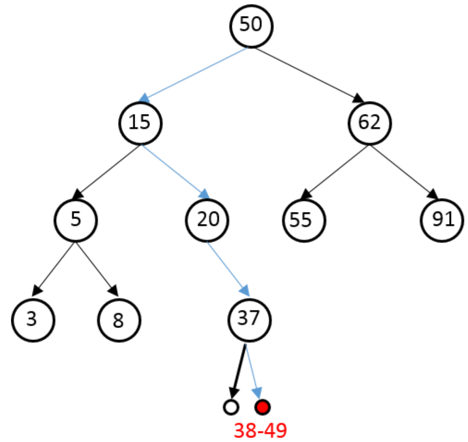
Hint: Assume the following member variables for each node: left_child, right_child and parent_node

SAMPLE SOLUTION (C code)

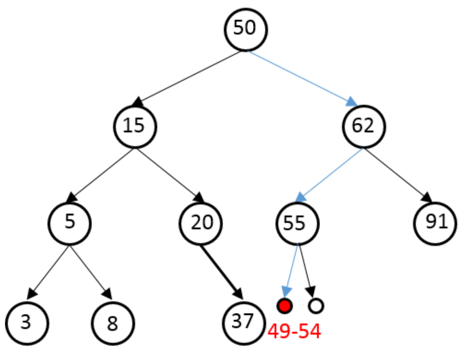
```
void function(node *n, bool  b, int &min, int &max){
    if (b==true) {
        if (n==NULL || n->left != NULL)
            exit(EXIT_FAILURE);
        else {
            max = n->value - 1;
            while (n->parent_node) {
                node *m = n->parent_node;
                if (m->right_child!=NULL && m->right_child == n) {
                    n = m;
                    break;
                } else
                    n = m;
            }
            if (max >= n->value+1) min = n->value+1;
            else min = -1 (or -INFINITY);
        }
    } else {
        if (n==NULL || n->right != NULL)
            exit(EXIT_FAILURE);
        else {
            min = n->value + 1;
            while (n->parent_node) {
                node *m = n->parent_node;
                if (m->left_child!=NULL && m->left_child == n) {
                    n = m;
                    break;
                } else {
                    n = m;
                }
            }
            if (min <= n->value+1) max = n->value-1;
            else max = -1 (or INFINITY);;
        }
    }
}
```



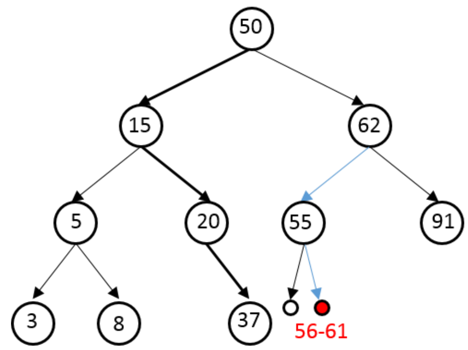
Case 1 (b=1)



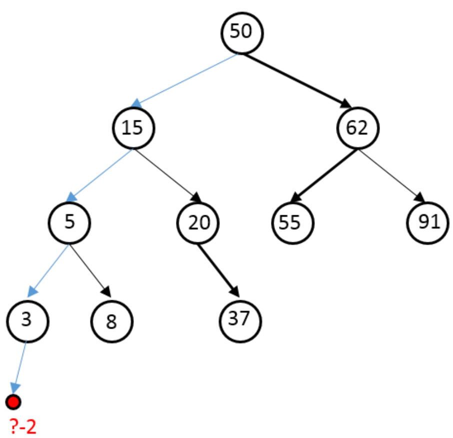
Case 2 (b=0)



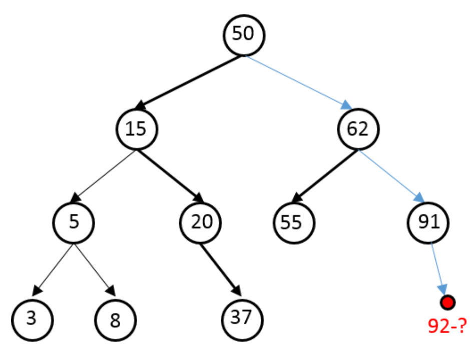
Case 3 (b=1)



Case 4 (b=0)



Case 5 (b=1)



Case 6 (b=0)

Question	Score
Q1	/20
Q2	/20
Q3	/20
Q4	/20
Q5	/20
Bonus	
Total	/100