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 …

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

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

```
MATRIX_MULTIPLY ($A[1 \dots n][1 \dots n], B[1 \dots n][1 \dots n]$)      Cost    Times
1.    for ($i = 1;\ i \le n;\ i$++) {                                      $C_1$
2.        for ($j = 1;\ j \le n;\ j$++) {                                  $C_2$
3.            $c_{ij} \leftarrow 0$                                        $C_3$
4.            for ($k = i; k \le j; k$++) {                                $C_4$
5.                $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$         $C_5$
6.            }
7.        }
8.    }
9.    return $C$                                                           $C_6$
```

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?

```
FUNCTION_B (int $n$)                                   Cost    Times
1.    int sum = 0;                                     $C_1$
2.        for ($i = 1;\ i \le n;\ i \mathrel{*}= 2$)   $C_2$
3.            for ($j = 0;\ j < i;\ j$++)              $C_3$
4.                sum++;                               $C_4$
5.    return sum                                       $C_5$
```

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?

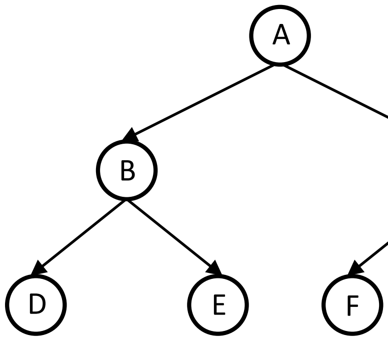
2. **Class Participation Type Algorithms (20 points total)**

   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.

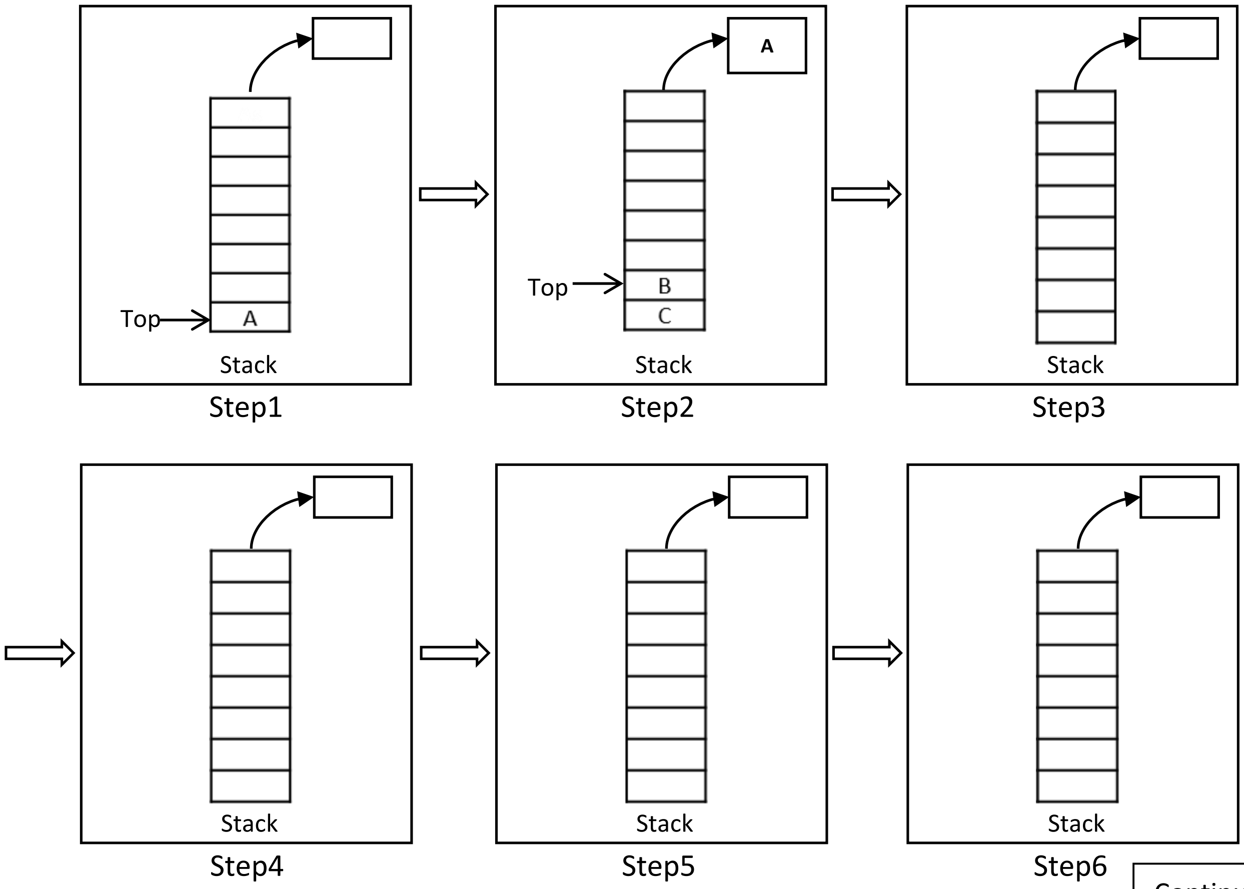
   b) **(10 points)** Given the array J = [1,2,3,4,5,6,7,8,9,10], explain in what order you would add the elements in J to create a binary search tree with minimal height.

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

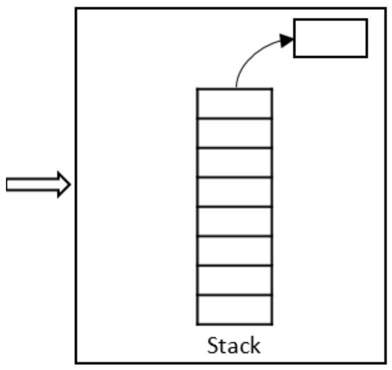
Consider the following binary tree:



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 3-7.

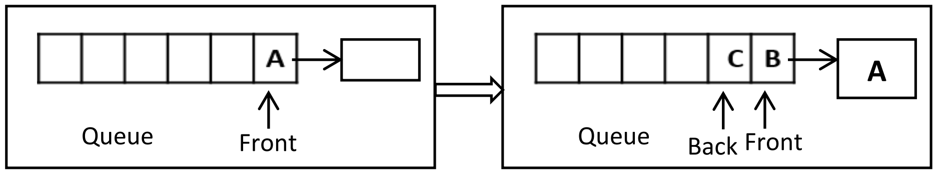


Continues on the next page.

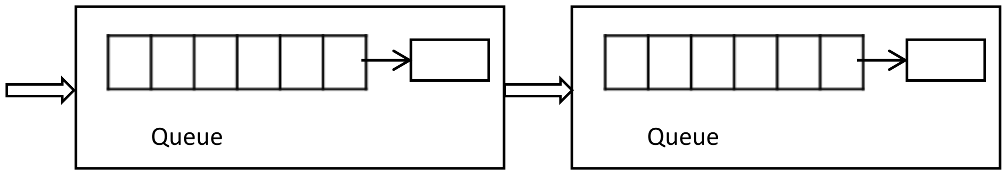


Step7

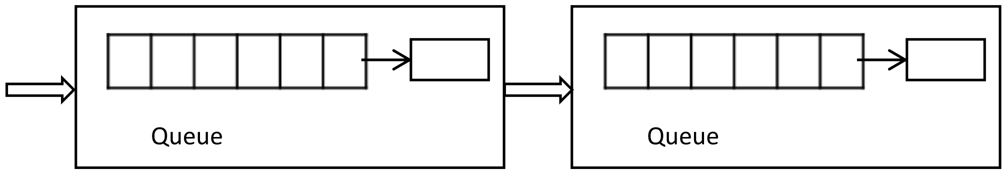
b) **(10 points)** Consider using a queue for **<u>Breadth-First Search</u>** (BFS) traversal of the given binary tree in the previous question. Whenever an element is dequeued from the queue, the element will be printed out and its children enqueued (if not NULL). Fill in the queue elements, update front & back, and write the dequeued elements in Steps 3-7.



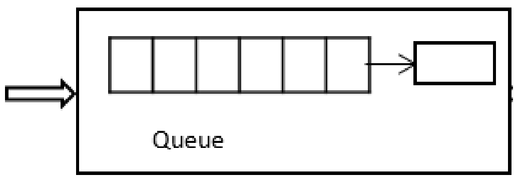
Step1 Step2



Step3 Step4



Step5 Step6



Step7

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

Consider the following condition C1:

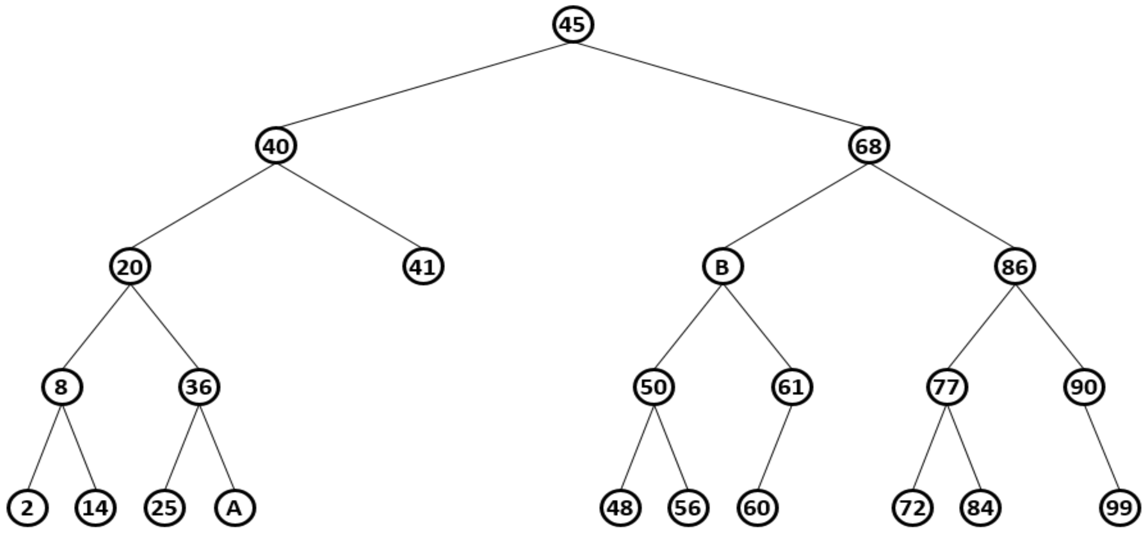
C1: Every node is either a leaf or a full node.

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

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

5. **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:

B:

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.

Continues on the next page.

c) (**Bonus Question – 10 points**) Write C or C++ code to implement a function that takes as input a pointer to a binary search tree node, a Boolean parameter, and two integers passed by reference. The function determines the range of values that an empty child position can take. If the bool is true then the left child position is considered, and otherwise the right child position is considered.

The outputs of the function are lower and upper bounds on the possible range of values and returned through the two integer inputs that are passed by reference. Assume any needed parameters.

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

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