General Instructions

1. You may use the class notes and homework assignments, or any handwritten material. No electronic devices are allowed.

2. The work is to be your own and you are expected to adhere to the UMass Boston honor system.

3. Write your answers in the available spaces, using the back of the page if needed. Write clearly and concisely and try to avoid cursive.

4. Please explain your answers if needed but do it briefly.

5. You may use any proof technique we showed in class or any other technique, as long as it constitutes a mathematical proof. Remember that a proof by example is generally good only to show that something is NOT true.

6. If you base your answer on a homework question state exactly which question it was.

Good Luck!

Name: ____________________________
1. (30%) **Medians and order statistics**: If we had an algorithm that finds the median of a sequence in linear time (worst case) – `findMedian(A, p, r)` which returns the index of the median of the sequence `A[p..r]`, describe a worst-case linear time algorithm that finds any order statistics. Provide a brief runtime analysis.
2. **Binary Search Trees:** (40%):

(a) (20%) Show how to sort a set of \( n \) numbers in \( O(n \log n) \) time using only the following binary search tree operations: Minimum, insert and successor. You may assume each operation takes \( O(\log n) \) time. Your initial input is just the array of numbers in random order.

(b) (20%) Let \( b_n \) be the number of all possible binary search trees with \( n \) nodes. Trivially, \( b_0 \) is 1. Show that \( b_n = \sum_{k=0}^{n-1} b_k b_{n-1-k} \).

**(Hint:** Don’t look for hidden catches. Think combinatorially. I’m looking for quite a simple explanation).
3. Dynamic Programming:

(a) (8%) See question Question 2b above. Draw all the possible trees for 2 and 3 nodes (assume
the keys are \{1, 2\} and \{1, 2, 3\}).

(b) (9%) The question can be solved using Dynamic Programming. Based on 3a, show how
many binary search trees can be constructed for 4 nodes by filling the table below (you can
and should use the formula in 2b which can be used as recursion and DP):

<table>
<thead>
<tr>
<th>n</th>
<th>b_1</th>
<th>b_2</th>
<th>b_3</th>
<th>b_4</th>
</tr>
</thead>
</table>

(c) (8%) What is the run time of the DP algorithm as a function of n? Explain briefly.