NAME: ____________________

CSE 421
Introduction to Algorithms
Sample Midterm Exam Fall 2014

DIRECTIONS:

- Answer the problems on the exam paper.
- You are allowed one cheat sheet.
- Justify all answers with proofs, unless the facts you need have been proved in class or in the book.
- If you need extra space use the back of a page
- You have 50 minutes to complete the exam.
- Please do not turn the exam over until you are instructed to do so.
- Good Luck!

1	/25
2	/25
3	/25
4	/25
Total	/100

1. (25 points, 5 each) For each of the following problems answer **True** or **False** and BRIEFLY JUSTIFY you answer.

   (a) $n^{2.1} = O(n^2 \log n)$.

   (b) There is a polynomial time algorithm for deciding whether a graph is bipartite or not.

   (c) If an undirected connected graph G has a unique heaviest weight edge e, then e cannot be part of any minimum spanning tree.

   (d) If all edges in a graph have weight 1, then there is an $O(m+n)$ time algorithm to find the minimum spanning tree, where m is the number of edges and n is the number of vertices.

   (e) If $T(n) \leq 10T(n/3) + n^3$, $T(1) = 1$, then $T(n) = O(n^3)$.

2. (25 points) A perfect matching of an undirected graph on $2n$ vertices is a matching of size n, namely n edges such that each vertex is part of exactly one edge. Give a polynomial time algorithm that takes a tree on $2n$ vertices as input and finds a perfect matching in the tree, if such a matching exists. HINT: Give a greedy algorithm that tries to match a leaf in each step.

3. (25 points) A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S. For instance, if S is

$$5, 15, -30, 10, -5, 40, 10,$$

then $15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not. Give a polynomial time algorithm that takes n numbers as input, and outputs the contiguous sequence of maximum sum.

4. (25 points) Given *sorted* array of n distinct integers, arranged in increasing order $A[1, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give an algorithm that runs in time $O(\log n)$ for this problem. HINT: Consider the algorithm that compares $A[\lceil n/2 \rceil]$ and $\lceil n/2 \rceil$, and uses that comparison to recurse on either the first half or the second half of the array. Prove that if $A[\lceil n/2 \rceil] > \lceil n/2 \rceil$, such an i cannot be in last $n - \lceil n/2 \rceil$ coordinates, and if $A[\lceil n/2 \rceil] < \lceil n/2 \rceil$, then such an i cannot be in the first $\lceil n/2 \rceil$ coordinates.