Name: _____

1	/25
2	/13
3	/12
4	/10
5	/20
6	/20

CSE 521: Design and Analysis of Algorithms
Final Exam
June 9, 1997

Instructions:

Give your answers in the spaces provided on these sheets. You need not justify your answer unless you are explicitly asked to. Feel free to describe an algorithm in English if that is more convenient than writing pseudocode.

There are 6 problems on this exam. You will have 80 minutes in total. Don't spend too much time on any one problem.

1. (25 points) In this problem you are interested in multiplying two polynomials $a(x)$ and $b(x)$ with integer coefficients, using the FFT and the evaluation-interpolation scheme. I will lead you through the steps. For your computational ease, $a(x)$ and $b(x)$ will each have degree 1.

   (a) Suppose each coefficient of $a(x)$ and $b(x)$ has absolute value at most 3. Why is $p = 37$ large enough so that you can do the whole computation in $\mathcal{Z}_p$, and still recover the correct answer over the integers?

   (b) Why must you evaluate $a(x)$ and $b(x)$ at $N \geq 4$ points rather than only 2?

   (c) Show that 6 is a primitive 4th root of unity in $\mathcal{Z}_{37}$.

   (d) Suppose $a(x) = 3 - x$ and $b(x) = 2 + 3x$. I will now provide you with the results of the evaluation and pointwise multiplication steps, and ask you to do the interpolation step:

$$\begin{aligned} a(6^0)b(6^0) &\equiv 10 \pmod{37} \\ a(6^1)b(6^1) &\equiv 14 \pmod{37} \\ a(6^2)b(6^2) &\equiv 33 \pmod{37} \\ a(6^3)b(6^3) &\equiv 4 \pmod{37} \end{aligned}$$

Use Fast Fourier Interpolation to find $a(x)b(x)$ as a polynomial in $\mathcal{Z}_{37}$. Show the parameters and returned values of every recursive FFT call, much as I did in class. (Hint: $6^{-1} \equiv 31 \pmod{37}$ and $4^{-1} \equiv 28 \pmod{37}$.)

(e) Recover $a(x)b(x)$ as a polynomial over the integers. Explain how you knew which coefficients to change without having to use the $\Theta(n^2)$ polynomial multiplication algorithm.

2. (13 points) Let $A[1..n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion* of A. Recall that an order-statistic tree is a red-black tree in which each node is augmented by a "size" field. Describe how to use an order-statistic tree to count the number of inversions in A in time $O(n \log n)$.

3. (12 points) Show the binomial heap that results when the node with key 28 is deleted from the binomial heap shown below. (Show some of your intermediate heaps, so that I can give you partial credit if you go awry.)

4. (10 points) Suppose that you wanted to implement an operation Increase-Key(H, x, k) for the binomial heap H. Given a pointer to the node x in H, this operation should increase x's key to k. Describe the most efficient worst-case algorithm for this operation that you can, and explain its asymptotic running time.

5. (20 points) Suppose $G = (V, E)$ is a flow network with source s, sink t, and capacity function c. A *cut* of G is a subset $S \subseteq V$ such that $s \in S$ and $t \in V - S$. Given a flow f in G, a cut S is *saturated* if and only if $f(S, V - S) = c(S, V - S)$; that is, every edge from S to $V - S$ is saturated, and every edge from $V - S$ to S has zero flow.

   (a) Suppose that a maximum flow f has already been found in G using a preflow-push algorithm. Describe a fast algorithm to output a saturated cut S, and state its asymptotic running time.

   (b) Explain why the value $f(s, V)$ of the maximum flow f is equal to the capacity $c(S, V - S)$ of this saturated cut. (You've just proved the difficult half of the famous "Max-flow min-cut theorem", which states that the maximum flow is equal to the minimum capacity of any cut.)

6. (20 points) You are given a complete undirected graph on the vertex set $V = \{a, b, c, d, e, f\}$ with the following edge cost matrix:

	a	b	c	d	e	f
a		14	15	10	17	12
b			8	11	9	9
c				15	7	7
d					20	14
e						13

This cost function satisfies the triangle inequality.

(a) Simulate Kruskal's algorithm to find a minimum cost spanning tree T of this graph. Each time an edge e is added to T, redraw the current spanning forest T and the full collection of Union-Find trees resulting from adding e to T. Use Weighted Unions, but without path compression.

(b) Using your answer to part (a), construct a Hamiltonian cycle of the vertices that approximates the optimal traveling salesman tour to within a factor of 2. (In order to do this in a unique way, please make node **b** the root of your spanning tree, and order the children of each node alphabetically.) What are the total costs of your spanning tree, and of your tour? (For your information, I believe the optimal tour has cost 54, but I wouldn't bet on it without spending exponentially more time.)