Homework 6
Fundamental Algorithms, Fall 2002, Professor Yap

Due: Wed Dec 11 (last recitation) or Thu Dec 12 (to one of us).

1. **Biconnected Component (20 Points)**
   Do parts (a) to (d) in Exercise 22-2, page 558 of CLR Text.

2. **Graph Diameter (20 Points)**
   Let $G = (V, E)$ be a bigraph, assumed to be connected. The **diameter** of G is $\max_{u,v \in V} \delta(u, v)$, where $\delta(u, v)$ is the length of the shortest path between u and v (what we also called "link distance" in class). Give an efficient algorithm to estimate the diameter within a factor of 2, i.e., your algorithm must return a number D such that the diameter of G lies in the interval $[D, 2D]$? Bound the running time.

3. **Dijkstra's Algorithm (5+20 Points)**
   Consider running Djikstra's algorithm on the graph in Figure 24.6 (page 596, CLR Text). However, instead of the weights there, you must add a positive integer $\Delta > 0$ to each weight. We want you to choose the smallest Δ such that the order in which nodes that becomes "known" is different than the original order, which is (s, y, z, t, x). You should try $\Delta = 1, 2, 3$, etc until you see a different order emerging.

   What to hand in: tell us what Δ is, and submit a table showing your simulation of Dijkstra. The table is rather similar to Prim's algorithm in the previous homework.

   CONVENTIONS: the data for each row of your table should correspond to this order: (s, t, x, y, z). The first row is $(0, 10 + \Delta, \infty, 5 + \Delta, \infty)$. To fill in the ith row, you first copy the SMALLEST weight in the $(i - 1)$st row that is still "unknown" and underline it. Then you proceed to fill in the rest of the rows (use double quotes (") to indicate a repeated value, and leave blank those entries corresponding to "known" nodes).

4. **Bellman-Form Algorithm (5+20 Points)**
   The Bellman-Ford algorithm detects negative cycles (p.588, CLR). Suppose you also want to know all those vertices that are in a negative cycle. How do you modify the algorithm? HINT: keep track of the shortest paths using the $\pi[u]$ array (cf.p.584).

5. **Shortest Path (20 Points)**
   Consider the min-cost path problem in which you are given a digraph $G = (V, E; C_1, \Delta)$ where C_1 is a positive cost function on the edges and Δ is a positive cost function on the vertices. Intuitively, $C_1(i, j)$ represents the time to fly from city i to city j and $\Delta(i)$ represents the time delay to stop over at city i. A jet-set business executive wants to construct matrix M where the (i, j)th entry $M_{i,j}$ represents the "fastest" way to fly from i

to j. This is defined as follows. If $\pi = (v_0, v_1, \ldots, v_k)$ is a path, define

$$C(\pi) = C_1(\pi) + \sum_{j=1}^{k-1} \Delta(v_j)$$

and let $M_{i,j}$ be the minimum of $C(\pi)$ as π ranges over all paths from i to j. Please modify the Floyd-Warshall Algorithm, to compute M for our executive.