This is an open-book test. You have approximately eighty minutes to complete it. You may consult any books, notes or other paper-based inanimate objects available to you. To avoid confusion, read the problems carefully. If you find it hard to understand a problem, ask us to explain it. If you have a question during the test, please come to the front or the side of the room to ask it.

This exam comprises 15% of the points on which your final grade will be based. Partial credit may be given for wrong answers. Your exam should contain six problems (numbered 0 through 5) on eight pages, with two more blank pages at the back of the exam. Please write your answers in the spaces provided in the test; in particular, we will not grade anything on the back of an exam page unless we are clearly told on the front of the page to look there.

Relax -- this exam is not worth having a heart failure about.

**Problem #0 - 1 point, 1 minute**
Put your name on each page. Also make sure you have provided the information requested on the first page.

**Problem #1 - 4 points, 10 minutes**
Suppose that the following statements initialize the union/find structure described in CLR section 22.3.

```c
for (k=0; k
```

On the next page show the data structures, including ranks of representative elements:

- Union (0,1);
- Union (2,3);
- Union (0,2);
- Union (4,5);
- Union (0,4);

Results of Union calls

<table>
<thead>
<tr>
<th>call</th>
<th>resulting data structure, including rank(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union(0,1);</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CS 170, Midterm #1, Spring 2000

Problem #1 - 4 points, 10 minutes

- Union(2,3);
- Union(0,2);
- Union(4,5);
- Union(0,4);
Problem #2 - 6 points, 15 minutes
This problem concerns an implementation of Prim's algorithm as given in CLR that maintains the priority queue as an unsorted singly linked list.

The three C declarations below are used to store the graph in adjacency list format. Some extra fields have been added that anticipate the application of Prim's algorithm.

```c
struct Graph {
    struct Vertex vertices[ ]; // The array of vertices
};

struct Vertex {
    int index;                  // The "name" of the vertex
    int key;       // Key value used to organize the priority queue
    struct QueueNode* qPtr;     // Pointer to a node in the queue
    struct NbrList* neighbors;  // Adjacency list.
};

struct NbrList {
    struct Vertex* nbr;         // The other endpoint of this edge
    int w;                      // The weight of this edge
    struct NbrList* next;
};

struct QueueNode {
    struct Vertex* v;      // Pointer to the corresponding vertex
    struct QueueNode* next; // Ptr to the next priority queue element
};
```

Elements of the priority queue, implemented as an unsorted singly linked list, are declared in C as follows.

```c
struct QueueNode {
    struct Vertex* v;      // Pointer to the corresponding vertex
    struct QueueNode* next; // Ptr to the next priority queue element
};
```

(The structs will all be classes in Java.)

Part A

Give an estimate of the worst-case running time required to build the spanning tree for a graph of n vertices and e edges.

Part B

Assume that the contents of the queue are initially V1, V2, V3, V4. Supply edge weights for the graph below that produce worst-case behavior.
Problem #3 - 3 points, 10 minutes

Give a tight estimate for the following recurrence, simplified as much as possible. Assume that values of $T$ for small values of $n$ are constant; that is, $T(0) = 1$, $T(1) = 1$, $T(2) = 1$.

$$T(n) = 9T(n/3) + n^2 + n \log n$$

Explain your answer.

Problem #4 - 8 points, 20 minutes

Prove that a graph $G = (V,E)$ with no isolated vertices is strongly connected if and only if there is a circuit in $G$ that included every edge at least once (and possibly more than once).

Problem #5 - 8 points, 24 minutes

Give an efficient algorithm that, given a directed acyclic graph $G = (V,E)$ and a vertex $a$ in $V$, counts the number of paths from $a$ to all other vertices. For example, there are five paths from $a$ to $b$ in the graph displayed below, namely $aeb$, $aeb$, $aceb$, $acedb$, and $acdb$. 
Provide sufficient comments for us to understand how your algorithm works. An incorrect algorithm may earn you partial credit if we can understand it; if you know it won’t work, provide a counterexample with your algorithm description.