Part B

1. Show that every tree with n nodes has exactly $n-1$ edges.
   (**Hint.** Induction on n, using the fact that every finite tree has a *leaf*, i.e., a node of degree one.)

   Solution: Basis Case: $n = 1$, so the tree consists of a single node, and hence it has no edges; $n - 1 = 1 - 1 = 0$ edges. Induction Step: suppose that we have a tree with $n + 1$ nodes. Pick a leaf and the edge that connects it to the rest of the tree. Removing this leaf and its edge results in a tree with n nodes, and hence—by induction hypothesis—with $n - 1$ edges. Thus, the entire tree has $(n - 1) + 1 = n$ edges, as required.

2. Show that a graph with n nodes and more than $n - 1$ edges must contain at least one cycle.

   Solution: We prove this by induction, with the Basis Case $n = 3$ (since a graph without multiple edges cannot have a cycle with less than 3 nodes). If $n = 3$, and there are more than $n - 1 = 2$ edges, there must be exactly 3 edges. So the graph is a cycle (a "triangle"). Induction Step: consider a graph with $n + 1$ many nodes ($n \geq 3$), and at least $n + 1$ many edges. If the graph has a node with zero or one edges adjacent to it, then by removing the node (and the edge, if there is one), we obtain a graph with n nodes and at least n edges, and so—by induction hypothesis—the resulting graph has a cycle, and so the original graph also has a cycle. Otherwise, all nodes have at least two edges adjacent to it. Suppose v_0 is such a node, and $(v_0, x), (v_0, y)$ are two edges. Remove v_0 from the graph, and remove the edges $(v_0, x), (v_0, y)$ and replace them by the single edge (x, y). Again—by induction hypothesis—there must be a cycle in the resulting graph. But then there must be a cycle in the original graph as well. (Note that there are $n + 1$ nodes, so after removing v_0 there are n nodes, and $n \geq 3$.)

3. Prove the **Exchange Lemma:** Let G be a connected graph, and let T_1 and T_2 be any two spanning trees for G. For every edge e in $T_2 - T_1$, there is an edge e' in $T_1 - T_2$ such that $(T_1 \cup \{e\}) - \{e'\}$ is a spanning tree for G.
   (**Hint.** Let e be an edge in $T_1 - T_2$. Then $T_2 \cup \{e\}$ contains a cycle—can all the edges in this cycle belong to T_1?)

   Solution: Let e be any edge in $T_2 - T_1$. We must prove the existence of $e' \in T_1 - T_2$ such that $(T_1 \cup \{e\}) - \{e'\}$ is a spanning tree. Since $e \notin T_1$, by adding e to T_1 we obtain a cycle (by the previous two questions!). A cycle has at least 3 edges (the graph G has at least 3 nodes, since otherwise it could not have two distinct spanning trees!). So in this cycle, there is an edge e' not in T_2. The reason is that if every edge e' in the cycle did belong to T_2, then T_2 itself would have a cycle. By removing e', we break the cycle but the resulting graph, $(T_1 \cup \{e\}) - \{e'\}$ is still connected and now acyclic, and therefore a spanning tree.

4. Recall the quick-union (slow-find) program presented in class:

```
#include <stdio.h>
#define N 10000
main()
{
    int i, j, p, q, t, id[N];
    for (i=0; i<N; i++)
      id[i]=i;
    while (scanf("%d %d\n", &p, &q)==2) {
      for (i=p; i!=id[i]; i=id[i]);
      for (j=q; j!=id[j]; j=id[j]);
      if (i==j)
        continue;
      id[i]=j;
      printf(" %d %d\n", p, q);
      }
}
```

Modify this program, by adding an array `sz[i]` which for each object with `id[i]==i`, maintains the number of nodes in the associated tree. Then, `sz` is used in the union operation so that the smaller tree is linked to the larger, thus preventing the growth of long paths in the tree.

Solution:

```
#include <stdio.h>
#define N 10000
main()
{
    int i, j, p, q, t, id[N], sz[N];
    for (i=0; i<N; i++) {
      id[i]=i;
      sz[i]=1;
      }
    while (scanf("%d %d\n", &p, &q)==2) {
      for (i=p; i!=id[i]; i=id[i]);
      for (j=q; j!=id[j]; j=id[j]);
      if (i==j)
        continue;
      if (sz[i]<sz[j]) {
        id[i]=j;
        sz[j]+=sz[i];
        }
      else {
        id[j]=i;
        sz[i]+=sz[j];
        }
      printf(" %d %d\n", p, q);
      }
}
```