1. Run-times

	Run-time
Kruskal's MST	
Dijkstra's	
Topological sort	

2. Graphs

a. Draw the following graph: V={a,b,c,d,e }
E={(a,b):2, (a,c): 5, (a,d): 14, (b,c): 4, (b,d): 12, (b,e): 1, (c,d): 7, (c,e): 1, (d,e): 9}
where (x,y):z represents an undirected edge between x & y with weight z.

b. Find a minimal spanning tree using Kruskal's algorithm.

c. Find the shortest path from a to each vertex using Dijkstra's algorithm.

d. Dijkstra's algorithm does not necessarily work when the graph possesses negative weights. Will Kruskal's algorithm work with negative weights? Why or why not?

3. Parallelism

a. Write pseudo-code for a fork/join program that takes an array of doubles and multiplies all the elements together. What work and span do your answers have?

b. Suppose that we decided that 0's should be ignored (that is, not multiplied in); how could you change your program to accomplish this.

c. Intuition suggests that by using more processors/computers, we can always reduce the running time of a parallelizable algorithm, such as summing up an array of integers. Explain at a high level – as though you were explaining to a non-computer-scientist – why we can't get array summing down to constant time despite how many processors we throw at the problem.

4. Concurrency

a. What is the difference between a race condition and a data race, as we've described them in class?

b. What are potential consequences of handling concurrency from too coarsely-grained locking? What about the potential consequences from too finely-grained locking?