1. In the graph below (with edges labeled with maximum edge capacities), show the maximum flow. Show each step of your algorithm, including the augmenting paths.

2. Go to csilm.usu.edu. Under CS3, select Minimal Spanning Tree. Select the Watch tab to observe various minimal spanning tree algorithms. Find the minimum spanning tree for the graph shown below using Kruskal’s algorithm. Show your work.

3. Using Prim’s algorithm, show the steps of finding the minimal spanning tree of the following graph. Start at A:
4. For each graph below, indicate whether there is
   a. a Hamiltonian tour
   b. an Eulerian tour
   c. both
   d. neither

5. For the following graph, show Num, Low, articulation points. [See algorithm 9.6 in section 9.6.2 of your text.] For ease in grading, start the depth-first numbering with node G.
6. Go to csilm.usu.edu. From CS3, go to Edge Labeling. Experiment with edge labeling as directed on the ILM.
   e. For a BFS, what kinds of edge labels can appear? [Change the type of search option to BFS.]
   f. What is the advantage of giving nodes a DFS number?
   g. If edges are undirected, what kinds of edge labels can appear?
   h. How could identifying back edges be of benefit?

7. For the graph below, label the edges as tree, forward, cross, or back. Start at node 15. For ease in grading, consider successor nodes in numeric order.

8. If we don't get this far in lecture on Wednesday, this problem will not be required. For the graph below, find the strongly connected components using two depth-first searches (as in section 9.6.5). Show each step. For ease in grading, start at node 1 and visit successors in numeric order. In restarting, start at lowest unvisited node.
   a. The post order traversal number.
   b. The reverse graph.
   c. The nodes in each strongly connected component.
Notes

Turn in your written homework through Eagle in a .doc, .odt, .pdf format or you can bring a paper copy of the homework with you to class. It will be graded by randomly selecting a subset of problems to evaluate. Not every problem will be graded. Bring a copy of the answers to class so that we can discuss them.

Written homework provides an excellent framework for achieving the goals of obtaining a working knowledge of data structures, perfecting programming skills, and developing critical thinking strategies to aid the design and evaluation of algorithms. Since programming has a high overhead in terms of program entry and debugging, all important topics in this course cannot be covered via programming projects. Written homework exercises allow students to learn important material without a high time investment. Although the point value is low, the benefits are great. You can perfect your programming skills without spending hours at the computer and can get feedback on your thinking skills from your study partners. Students that consistently do quality homework, have far superior test scores. Because assignments are done as a group and any questions are discussed in class or during office hours, written solutions to the homework will not be provided.

Note, these exercises may be done in groups of one, two, or three. If more than one person is involved, list all the names on one set of answers. Groups may change throughout the semester. Answers should not be compared with others not in your group. You will learn much more by working in a group than you will learn working by yourself.