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. Turn in only one copy of the assignment for all group members. 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.*

1. For the graph below for buying a home, create an event graph (by making the nodes arcs and adding dummy arcs) show the critical path and the slack on each edge.

2. In the graph below, show the maximum flow. Show each step of your algorithm, including the augmenting paths.
3. Find the minimum spanning tree for the graph shown below using Kruskal’s algorithm. Show your work.

4. Using Prim’s algorithm, show the steps of finding the minimal spanning tree of the following graph. Start at A:

5. For each graph below, indicate whether there is 
   (a) a Hamiltonian tour
(b) an Eulerian tour
(c) both
(d) neither

6. For the following graph, show Num, Low, articulation points. For ease in grading, start the depth-first numbering with node G.

7. For the graph below, label the edges as forward, cross, or back. Select vertices as roots of the forest in alphabetical order.
8. For the graph below, find the strongly connected components using two depth-first searches (as in section 9.6.5). Show each step. The post order traversal number. The reverse graph. The nodes in each strongly connected component.