1. (10 points) Problem 5.4 from your textbook.

2. (10 points) Problem 5.9 from your textbook.

3. (10 points) Problem 5.10 from your textbook.

4. (10 points) In the [Kumar, 1992] paper we encountered a particular way of defining the tree width of a constraint graph. Read up on tree width in Section 5.4 of your textbook. Investigate if the two notions are compatible. If so, explain how. If not, explain why not.

5. (10 points) The GPC website (http://www.cs.vt.edu/~gpc) is a formidable one presenting rules and regulations for all forms of graduate study. Pick a program of study (e.g., M.S. thesis option, M.S. coursework option, or Ph.D.), scour the site to identify requirements of that program, and model it as a CSP. How many variables are there? What are their domains? What is the maximum arity of the constraints? What approach would be suited for solving these CSPs?

Having gained experience modeling one graduate program as a CSP, which graduate program do you think will boast the lowest tree-width? Is the lowest tree-width correlated with the number of ways in which the requirements for the program can be fulfilled? (Watch out: this question is tricky!)

6. (50 points) In this exercise, you are going to code up a Sudoku solver that uses constraint satisfaction techniques.

First, carefully define the variables, domains, and constraints of this problem. Keep in mind there are numerous formulations and there is no single answer. We are only looking to see if you have captured the Sudoku puzzle in your formulation, not if you got the formulation we have in mind.

Next, you will implement four programs that will perform varying degrees of search and constraint propagation (i.e., simplification of domains):

- The first program you will implement is a mere backtracking search (i.e., depth-first search). Keep in mind that the algorithm was covered way back in Chapter 3 and is also reviewed in Section 5.2.

- The second program conducts what is called ‘forward checking.’ We did not cover this in class but it is a minor variant of maintaining arc-consistency. Here, after we take a step down the search tree (e.g., by placing a tile in a specific position), we ‘flush’ out to find all variables that get constrained as a result of this step, and reduce the domains of these variables. We backtrack when a domain gets empty. Therefore, forward checking is performed only with respect to the current step. Another way of saying this is we can perform arc-consistency on only those edges $(V_i, V_j)$ where $V_i$ is the just set variable.
• The third program maintains arc-consistency, meaning it performs a full lookahead. It extends constraint propagation to even the variables that have not been assigned. For instance, after you have set variables $V_1$, $V_2$, $V_3$, and $V_4$, maintaining arc-consistency might simplify the domain of position $V_7$ because for some values of this position, there is no possible value for variable $V_8$. Use the AC-3 algorithm (Fig. 5.7) for implementing this (and perhaps the previous) approach. Once again, you backtrack when a domain becomes empty.

• The final program uses the AllDiff constraint checker. Again, we haven’t covered this in class but read up ‘Handling special constraints’ in Section 5.2 to see what this is about.

Notice that all programs except the first perform constraint propagation at each step of search – it is the level of constraint propagation that is different.

Using a range of problems from easiest to most difficult, explore how the four programs fare. Which of them runs fastest? Time each carefully, taking care to distinguish the time spent in constraint propagation from that spent in search - i.e., ‘thinking’ versus ‘doing.’ Make interesting observations about the search times, the effectiveness of the different strategies, and the range of problems that each strategy can solve.

For full credit, give us an URL where your code can be checked (and downloaded, if necessary), write up a summary of how you represented the states, operators, the specific manner in which you did your experiments, and graphs/tables depicting the numbers you measured. Give English statements summarizing the conclusions and lessons learned.