1. **CSP Problem Formulation [15 points]**

Suppose that the Home Decorating show asks you for consulting to help out their designer. The problem is to decorate a room in a home. You must paint the walls, choose new furniture, choose art work and choose a rug. Paint comes in different colors and different tones, furniture comes in different colors and styles, artwork comes in different styles and tones, and rugs come in different colors and textures. The colors of the walls, furniture and rugs should complement each other. The tones of the paint and artwork should match and the styles of the furniture and artwork should match (note, matching doesn’t have to mean the values are the same, you can define match however you like. also note that the possible values for style of furniture might be different than the possible values for styles of artwork).

- Define the variables in this problem.
- Define the possible values for the variables (make these up – I suggest not making the domains too large, perhaps 3 or 4 choices).
- Define the constraints. You may do this in a number of ways, by listing legal combinations, or by simple pseudo code. The important thing is to be specific enough that for any pair of variables we can determine if some assignment of values is consistent.
- Draw the constraint graph.
- Give an example of a consistent assignment to the variables and an inconsistent assignment.

2. **CSP Problem Formulation [15 points]** Returning to your eStudent agent, describe a task that is relevant to a student that can be described as a CSP.

Give a few sentences describing the problem in English, then describe the problem formally as a CSP. Note that your task here is to formulate the problem, not solve the problem.

- Consider solving this problem using local search. Discuss how you would solve it. Do you think this would be a good approach? When is using local search a good idea?

3. **CSP Problem Solving [15 points]** Russell & Norvig 5.6, p. 159. Clearly show your work.

4. **CSP Problem Solving [15 points]** Russell & Norvig 5.8, p. 159. Clearly show the steps the AC-3 algorithm takes.

5. **[20 points] Minimax Search**

In this problem, you will run through the Minimax search strategy on the game of Tic-Tac-Toe. For a given initial position, the procedure generates a game tree where the current player is the first to move, and the players then alternate. At each node, the set of children corresponds to the set of non-symmetric moves that the current player can make.
The Minimax procedure relies on a heuristic function to determine the value of a particular state in the game. One heuristic function that can be used for Tic-Tac-Toe is the following:

$$Eval = 3 \cdot X_2 + X_1 - (3 \cdot O_2 + O_1)$$

where,

$X_2 =$ the number of rows, columns, or diagonals with 2 X’s and no O’s;

$X_1 =$ the number of rows, columns, or diagonals with 1 X and no O’s;

$O_2 =$ the number of rows, columns, or diagonals with 2 O’s and no X’s;

$O_1 =$ the number of rows, columns, or diagonals with 1 O and no X’s;

Starting from the initial position (the empty board with X to move), draw the game tree for depth two (one move for each player). Evaluate the leaf nodes of the tree using the heuristic function, then use Minimax to find the value for the root node. Circle the move that the root node player should take.

Note: If a node has two expanded states that are symmetric, you should only expand one of the similar states. So the first level will have 3 nodes.

6. **Alpha Beta Search [20 points]**

Alpha-Beta pruning can use the Minimax procedure described above to build a similar search tree, but it limits the search space by pruning nodes that will definitely not be reached in this game tree (assuming that the subtree is the whole game and that both players are playing optimally). When asked to perform Alpha-Beta pruning in this problem, indicate the nodes that will be pruned. Additionally, for each pruned node, specify the kind of pruning being performed (alpha or beta), and give the corresponding alpha and beta values.

(a) **Simple Alpha-Beta Pruning**

Consider the Tic-Tac-Toe game tree that you created in the previous example. Assume that we expand the nodes in the optimal order (i.e., the order that would enable alpha-beta pruning to prune the greatest number of nodes). Re-order the graph so that left-to-right node-expansion is the optimal ordering, and perform Alpha-Beta pruning.

(b) **Alpha-Beta Pruning on a Larger Tree**

In this second tree, fill in the values for the internal nodes using the given values at the leaves and the Minimax procedure, as before. Perform Alpha-Beta pruning twice on this tree, first assuming left-to-right node expansion, and second assuming right-to-left node expansion. In this tree, the root node is the maximizer player.

Begin by first giving a clear description of the components of the search problem for this game, then do 6.3a - 6.3d.