Discussion Problems:
"Constraint Programming Solvers"
Declarative Methods (JHU 600.432/632)
Prof. Jason Eisner

1. [12 points] This question is based very loosely on the murder mystery game *Clue* (known in some countries as *Cluedo*). Here are three variables and their current domains:

Murderer	**Weapon**	**Room**
Miss Scarlet	Rope	Billiard Room
Professor Plum	Lead Pipe	Study
Colonel Mustard	Revolver	Kitchen
Mrs. Peacock	Candlestick	

Apply arc-consistency to cross out values from the above domains. (Do not cross out impossible values unless arc-consistency would *discover* that they are impossible!) The tables list the value pairs that *are allowed* by each binary constraint.

Murderer	**Weapon**
Miss Scarlet	Rope
Miss Scarlet	Lead Pipe
Colonel Mustard	Lead Pipe
Colonel Mustard	Revolver
Colonel Mustard	Candlestick
Mrs. Peacock	Lead Pipe

Murderer	**Room**
Miss Scarlet	Study
Miss Scarlet	Kitchen
Professor Plum	Billiard Room
Professor Plum	Study
Colonel Mustard	Study
Mrs. Peacock	Study
Mrs. Peacock	Kitchen

Weapon	**Room**
Lead Pipe	Billiard Room
Rope	Kitchen
Rope	Study
Revolver	Kitchen
Candlestick	Kitchen

2. Consider these linear inequality constraints on integer variables:

```
X + Y + Z #> 3
2X + 3Y + 4Z #< 20
```

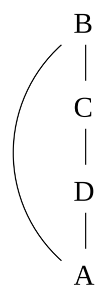
   (a) [3 points] Suppose all three variables initially have domain $-\infty..\infty$. Now you restrict Z to the domain `0..2`. You propagate this restriction through the two constraints above, using bounds propagation. What happens to the domains of Y and X?

(b) [4 points] Join the two constraints given above and project the result onto Y and Z. What is the new inferred constraint on Y and Z?

(c) [6 points] What would the answer to part (a) be if you did bounds propagation on all constraints (including the new constraint)?

3. Here is a simple system of constraints on 4 variables, shown alongside its constraint graph:

```
A*2 #= B,
B*2 #= C,
C*2 #= D,
A+D #= 27.
```



The unique solution is `[A,B,C,D] = [3,6,12,24]`.

(a) [10 points] Obtain this solution by the "variable elimination" algorithm (also known as "adaptive consistency"). We recommend that you eliminate variables in the order B, C, D. After each elimination step, show the modified set of constraints and the modified constraint graph:

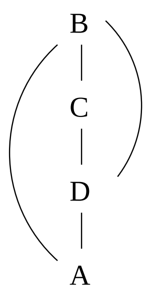
- Eliminate `B`:

- Eliminate `C`:

- Eliminate `D`:

- How would you use this sequence of constraint programs to obtain a complete solution to the original constraint program?

(b) [4 points] Suppose we had started with a different system whose constraint graph was as below (note the added edge from B to D). Then what would the modified constraint graph look like after eliminating B? Would the modified constraint program still involve only binary constraints?



4. Given a constraint program consisting of constraints $\{C_1, C_2, C_3, \ldots\}$.

   (a) [2 points] Suppose you join C_2 and C_3 to obtain a new constraint, C_{23}. You add C_{23} to the program. Now you (MUST, MAY, MUST NOT) delete C_2 and C_3 from the program in order to leave the set of solutions unchanged. *(circle one)*

   (b) [3 points] Suppose C_4 says `Y #= X*X+1` and C_5 says `Y #< 0`. These constraints are incompatible. How long will a basic backtracking solver take to detect that the problem is UNSAT? *(in each case, circle all that apply)*

      i. with no propagation:
      COULD BE IMMEDIATE, COULD TAKE EXPONENTIAL TIME

      ii. with arc consistency:
      COULD BE IMMEDIATE, COULD TAKE EXPONENTIAL TIME

      iii. with bounds consistency:
      COULD BE IMMEDIATE, COULD TAKE EXPONENTIAL TIME

   (Assume that all variables have domain `-99..99`. "Exponential time" means exponential in the total number of variables n, while "immediate" means constant with respect to n.)

   (c) [4 points] You are solving a standard graph coloring problem with k colors available. Recall that the two endpoints of each edge must be colored differently. Hence, the problem is UNSAT if there exists a clique of $k+1$ vertices. What standard constraint-solver technique would be guaranteed to detect that condition immediately?
   (*Hint:* For large k, detecting the condition is hard.)