Part II: Programming (100 points)

In this problem, you will implement a backtracking CSP solver that takes exactly three arguments from the command line:

1. A `.var` file that contains the variables in the CSP to be solved and their domains. Each line of the file contains a variable (represented by a single letter), followed by a colon and its possible values, each of which is an integer. For instance, the line "A: 1 2 3" indicates that the possible values for variable A are 1, 2, and 3. Note that there is a space separating the domain values.

2. A `.con` file that contains the constraints. Each line corresponds to exactly one constraint, which involves two variables and has the form `VAR1 OP VAR2`. `VAR1` and `VAR2` are the names of the two variables involved, and `OP` can be one of four binary operators: = (equality), ! (inequality), $>$ (greater than), and $<$ (less than).

3. The consistency-enforcing procedure, which can take one of two values: `none` and `fc`. If `none` is used, no consistency-enforcing procedure is applied, and the solver simply uses backtracking to solve the problem. `fc` indicates that the solver will use forward checking to enforce consistency.

Note:

- Whenever the solver needs to choose a *variable* during the search process, apply the most constrained variable heuristic, breaking ties using the most constraining variable heuristic. If more than one variable remains after applying these heuristics, break ties alphabetically.

- Whenever the solver needs to choose a *value* during the search process, apply the least constraining value heuristic. If more than one value remains after applying this heuristic, break ties by preferring smaller values.

Your program should allow exactly three arguments to be specified in the command line invocation of your program, in the order in which they are listed above. Sample input files will be available in the assignment page of the course website. Your program should write to `stdout` the first 30 branches visited in the search tree (or stop when the solution is reached). By branches we mean an assignment that violates a constraint or that leads to one or more unassigned variables having empty domains (when using forward checking). Write them in text form with each branch on one line. For example, suppose we had variables X and Y with domains $\{0, 1, 2\}$, variable Z with domain $\{1, 2\}$, and constraint Y=Z. The branches visited in the search tree without forward checking should be written as:

1. Z=1, X=0, Y=0 failure
2. Z=1, X=0, Y=1 solution

However, if forward checking is applied, the output should be:

1. Z=1, Y=1, X=0 solution