Project Write up for Constraint Satisfaction Problem Solver

* Compilation and Running
  + This project was built using eclipse and Java 7.
    - You can import the project using the standard dialog in eclipse.
  + To run, use the command: “java –jar inputFile.txt output.txt”
* Description of our approach
  + Our approach to designing a solution for this project is the “fail fast to succeed sooner” method. Essentially we tried everything at first to see what would work. Eventually we decided to implement Backtracking first, because it was the easiest and most readily produced.
    - Backtracking ended up revealing some serious flaws in our codebase however. We realized that we had not designed objects and methods that make this task an easy one, but we were able to make changes and get it working nonetheless.
    - One thing we noticed is that when we got back a bad inputs our program would have to try every possible combination of bags and items, which is on the order of millions for some problems.
  + Next we implemented Minimum Remaining Values + Degree Heuristic
    - The way we implemented this was relatively easy. The point of MRV + DH is to choose the next variable you are going to assign to an assignment intelligently to reduce the number of overall steps you’ll have to take.
    - Specifically we implemented MRV first to check for the most constrained variable in our domain. That is, the item when placed under the constraints succeeds the least number of times. When there was a tie that needed to be broken we used the Degree Heuristic.
    - The Degree heuristic for us was a simple measure of the number of constraints each MRV variable was subject to. We would then select the one with the highest degree
  + Then Least Constraining Values
    - We implemented least constraining values much in the same way that we did MRV, just with the difference being we were looking for the one that the least number of constraints on it. (So it was basically the reverse of MRV.) This performed a similar number of comparisons and calculations as MRV + DH.
  + Then Forward Checking
    - We really struggled with FC. Even with some directed help from friends, this was simply a difficult concept.
    - Essentially we tried to use the fact that we knew what type of constraint each item was subject to and tried to implement the forward pruning algorithm from there. Although this wasn’t very successful.
* Overall
  + The strategy we used:
    - Open file
    - Read in vars/bags/constraints
    - Depending on the approach, select the correct algorithm
      * All algorithms depend on Backtracking, which is most commonly used. Depending on the options given to BT, we can add FC, MRV + DH, and LCV
    - Generate Solution
    - Print Results
* Testing
  + We ran all of the tests included in the test files zip file.
    - Backtracking took by far the longest, and for some of the larger inputs we simply had to cut it off because it ate up all of the computer’s memory.
    - MRV + DH, LCV, and FC work better on the larger inputs than Backtracking (obviously.)
  + We also used the tests in the test folder to check to make sure our constraints were being properly satisfied. This ran on the Junit test framework.
* All other files like sample output and statistics can be located in the project’s home directory.