15-780 Homework 3
Deadline: 10:30 am on March 3 (Tuesday)
There are 240 total points: point values are listed with each question.

1) Please do problems 4.1-4.7 from the Russel/Norvig textbook (20 pts each).

Note: For problem 4.2 assume that $h(n)$ is admissible.

2) (100 pts total) In assignment 2 you implemented the Davis-Putnam-Logemann-Loveland (DPLL) algorithm for solving boolean satisfiability (SAT) problems. In addition to the basic algorithm you implemented a number of heuristics including: backtracking, unit propagation, and variable ordering heuristics. In this assignment you will build on your previous work by adding conflict driven clause learning and non-chronological backtracking to your SAT solver. These heuristics significantly improve the efficiency of SAT solvers and form the foundation of "second generation" SAT solvers including GRASP, Chaff, and BerkMin.

For this assignment, you should read the papers and slides under "February 5th" in the lecture portion of the website:
http://www.cs.cmu.edu/∼sandholm/www/cs15-780S09/schedule.html

The SAT problems required for this following questions can be found in the Homework 3 archive:
http://www.cs.cmu.edu/∼sandholm/www/cs15-780S09/hws.html
To check your solutions for each SAT problem, you may again use the C verifier provided in Homework 2.

a) (80 pts) Add conflict driven clause learning and non-chronological backtracking to your DPLL solver. Try one or more of the variable ordering heuristics described in the readings or feel free to invent your own. As in Homework 2, your implementation should be in Java or C/C++. For this problem you may either build on the provided solution to Homework 2 Problem 3 (to be posted once everyone has submitted Homework 2), or your own Homework 2 implementation. If you have any questions about the suitability of your Homework 2 implementation as the basis for this problem, email Byron.

Do *not* attach your code to your writeup. Instead, email your source code to Byron by the due date and time.

b) (20 pts) Run your program on each of the following SAT problems from the Homework 3 archive:

problem1.cnf
problem2.cnf
problem3.cnf
problem4.cnf
problem5.cnf

Report the amount of time that it takes to solve each of these SAT problems with the Homework 2 implementation and with your new implementation from part a. Although you *do not* have to report the actual satisfying assignments for this part, your program *does* need to produce solutions when they exist. The solutions generated by your implementation will be used to verify the correctness of your SAT solver.