# Problem set 6: SAT Solving

Exercises are meant to be solved collaboratively during the tutorial. Projects have to be handed in.

**Project 3: Extending your DPLL-Solver** The goal of this project is to extend your DPLL-Solver from Project 2 to a CDCL-Solver. The rules regarding languages and team size are therefore the same as in project 2. There are four parts to this project:

1. **Compulsory part**: Implement CDCL. This includes:
   - The 2-watched literals data structure for efficient unit propagation.
   - Deriving an asserting conflict clause on every conflict, and adding it to the clause database. You can use a learning schema of your choice.
   - Non-chronological backtracking, based on the assertion level of the conflict clause.
   - A clause deletion strategy of your choice.
   - A branching heuristic suitable for CDCL, such as VSIDS or VMTF.
2. **Bonus part**: Implement restarts. Experiment with phase saving and different restart policies.
3. **Bonus part**: Implement some preprocessing techniques, e.g., the ones presented in the lecture.
4. **Bonus part**: Implement proof logging in the DRUP (or DRAT) format. Your proofs must be verifiable with DRAT-trim.

Add a command line option to your solver to enable proof logging, and write the proof to a file. Make sure that your proofs justify any preprocessing that your solver performs, or else disable preprocessing steps not justified when generating them. Proof logging will be covered in the lec ture in Chapter 7.

Optionally, benchmark your solver against inputs used at the annual SAT competition. We recommend using inputs of older competitions, such as 2006 to 2010, because the time and memory limits (and difficulty of the problems) in more recent years are pretty high. Solutions can be submitted until March 21, 2024. You are asked to fork your git repository and provide us the link. Please add a short README file, explaining how to build/run your program.

**Exercise 1 :**

If a CDCL solver using the 2-watched literal data structure learns a conflict clause, it has to set two literals in it as watched literals. Describe what literals have to be set as the watched literals, in order for backtracking to work properly with the added clause.

**Exercise 2 :**

Some of the preprocessing techniques introduced in the lecture only preserve satisfiability of the formula, but satisfying assignments for the simplified formula might not satisfy the original one. Therefore, at the end a satisfying assignment for the original formula needs to be constructed, given a solution to the simplified formula. Decribe how an algorithm would do that for the following preprocessing techniques. What information needs to be kept for this purpose from preprocessing?

(a) Variable Elimination Resolution (VER), like NiVER

(b) Blocked Clause Elimination (BCE)