# DM550 Introduction to Programming
# Fall 2017 Re-re-exam Project (Java)

Department of Mathematics and Computer Science
University of Southern Denmark

July 2, 2018

**Introduction**
The purpose of this project is to try in practice the use of programming techniques and knowledge about the programming language Java. Please make sure to read this entire note before starting your work on this part of the project. Pay close attention to the three sections below.

**Exam Rules**
This project is part of the re-re-exam for DM550. To pass the re-re-exam this Java project (or the one from the ordinary exam or the re-exam) and a Python project (from the re-re-exam or from the re-exam or form the ordinary exam) have to be passed in order to pass the overall exam. This project hast to be done individually.

**Deliverables**
A short project report (at least 6 pages without front page and appendix) contain the following 6 sections has to be delivered:

- **front page** (course number, name, section, date of birth)
- **specification** (what the program is supposed to do)
- **design** (how the program was planned)
- **implementation** (how the program was written)
- **testing** (what tests you performed)
- **conclusion** (how satisfying the result is)
- **appendix** (complete source code)

The report has to be delivered as a single PDF file electronically using Blackboard's SDU Assignment functionality. No printed copies are required. **Do not forget to include the complete source code!**

# Deadline

August 26, 2018, 23:59

Late deliveries cannot be accepted, so please plan your time accordingly.

# The Problem

Your task in this part of the project is to write a SAT solver. A SAT solver is a program that reads a propositional formula and determines whether there is satisfying assignment of its variables. The propositional formula is represented in conjunctive normal form (CNF), i.e., as a conjunction of disjunctive clauses. Here, each clause is a disjunction of literals. Each literal is either a Boolean variables or a negated Boolean variable.

**Example 1:** The propositional formula $(X_1 \vee \neg X_2) \wedge \neg X_1$ is in CNF.

An assignment of variables to *true* and *false* satisfies a formula in conjunctive normal form if it satisfies all clauses. An assignment satisfies a clause if it satisfies at least one of its literals. An assignment satisfies a variable if it assigns it to *true*, and it satisfies a negated variable if it assigns it to *false*.

**Example 2:** The formula from Example 1 is satisfiable. The only satisfying assignment is the one that assigns both $X_1$ and $X_2$ to *false*. If we add the clause $X_2$ to the formula, we obtain $(X_1 \vee \neg X_2) \wedge \neg X_1 \wedge X_2$, which is not satisfiable. There are four possible assignments (both variables assigned to *true*; $X_1$ assigned to *true* and $X_2$ assigned to *false*; $X_1$ assigned to *false* and $X_2$ assigned to *false*; both variables assigned to *false*). The first two do not satisfy the clause $\neg X_1$ while the remaining two do not satisfy the clause $X_2$.

# The Input

For input to your program, the formulas in CNF are represented in a simplified DIMACS format. At the beginning of the file there can be lines starting with the letter "`c`". These have to be ignored as they contain only comments. Then there is a line starting with the letter "`p`", which defines the characteristics of the formula.

**Example 3:** `p cnf 2 3` signifies that the CNF has two variables $X_1$ and $X_2$ and consists of three clauses.

Each line following the "`p`" line contains one clause. The literals in a clause are represented as integers, where $i$ represents the variable $X_i$ and $-i$ represents the negated variable $\neg X_i$. Each line is terminated by 0.

**Example 4:** The clause $X_1 \vee \neg X_2$ would thus be represented as `1 -2 0`.

**Example 5:** The unsatisfiable formula from Example 1 could be written as follows:

```
c Example 1
p cnf 2 3
1 -2 0
-1 0
```

**Example 6:** The unsatisfiable formula from Example 2 could be written as follows:

```
c Example 2
p cnf 2 3
1 -2 0
-1 0
2 0
```

The home page of the course contains a number of possible inputs to test your program on.

# The Output

The output of your solver is at least one line with either "`s UNSATISFIABLE`" or "`s SATISFIABLE`". If a formula with $n$ variables is satisfiable, a line with a satisfying assignment should be output. This line starts with the letter "`v`" and lists $n$ integers. The first integer is either 1 (if $X_1$ is assigned to *true*) or $-1$ (if $X_1$ is assigned to *false*), the second 2 or $-2$, and so on.

**Example 7:** The output for the input from Example 6 should be as follows:

```
s UNSATISFIABLE
```

**Example 8:** The output for the input from Example 5 should be as follows:

```
s SATISFIABLE
v -1 -2
```

# The Tasks

The following tasks are meant only as a guideline. Feel free to use a different design.

1. Implement a method `readCNF` that reads a CNF in the simplified DIMACS format and stores it in an appropriate data structure. Hint: consider to use collection classes for storing e.g. a list of lists.

2. Implement a method `solve` that takes a CNF in your internal representation and outputs either `null` (if the formula is unsatisfiable) or a variable assignment satisfying the formula (if the formula is satisfiable). Hint: The easiest way is to use a generate-and-test appraoch, going systematically through all possible variable assignments. There are $2^n$ assignments for $n$ variables.

3. Implement a method `printResult` that prints the output in the format described above.