# Lgica en la Informtica / Logic in Computer Science
# June 20th, 2017. Time: 2h30min. No books or lecture notes.

**Note on evaluation:**
**eval(propositional logic) = max{ eval(Problems 1,2,3), eval(partial exam) }.**
**eval(first-order logic) = eval(Problems 4,5,6).**

**1** Consider the at-most-one (AMO) constraint, expressing that at most one of the propositional variables $x_1 \dots x_n$ is true, also written $x_1 + \cdots + x_n \leq 1$. Consider:
1) the encoding for AMO you know that needs the smallest (in terms of $n$) number of clauses, and
2) the encoding that needs the smallest number of auxiliary variables.
For each case, write **giving no further explanations**: a) the name of the encoding, b) which, and how many, auxiliary variables it uses, c) which, and how many, clauses (always expressing how many in terms of $n$).

**2** My friend John says that he has found a new way to speed up SAT solving. Before starting his SAT solver, he removes from the set of clauses $S$ some clauses he calls "unnecessary":
A: if there is some variable $x$ that appears only in positive literals of clauses of $S$, then he removes from $S$ all clauses containing $x$
B: similarly, if some variable $y$ appears in $S$ only in negative literals then he removes from $S$ all clauses containing $y$.
Note that after eliminating some "unnecessary" clauses, step A or B may be (or become) applicable for other variables, so John continues doing this until no more variables of type A or B exist and then launches his solver on a (hopefully) much smaller set of clauses. Is John's idea correct? Explain why, **in very few words**.

**3A:** What is the complexity of 2-SAT? (just answer, no explanations needed).
**3B:** Any set of propositional *positive clauses*, that is, clauses with only positive literals (no negations), is of course satisfiable, because the interpretation making all variables true is a model. What is the complexity of deciding the satisfiablity of a given "2-or-positive" set of clauses $S$, that is, such that every clause in $S$ is either positive or two-literal (or both)? Explain why, **in very few words**.
Hint: with two-literal clauses we can express that one variable is the negation of another variable.

**4:** Consider the following Prolog program and its well-known behaviour:

```
brother(joan,pere).
father(enric,joan).
uncle(N,U):- father(N,F), brother(F,U).

?- uncle(X,Y).
X = enric,
Y = pere.
```

Express the program as a set of first-order clauses $P$ and prove that $\exists x\, \exists y\ uncle(x, y)$ is a logical consequence of $P$. Which values did the variables $x$ and $y$ get (by unification) in your proof? **Only write the steps and values. No explanations.**

**5:** For each statement, say whether it is true or false and show why **in an as simple and short as possible way**:
**5A:** The formula $\forall x\, \exists y\ (p(x, f(y)) \wedge \neg p(x, y))$ is satisfiable.

**5B:** $\forall x\, \forall y\, \exists z\ q(x, z, y) \models \forall x\, \exists z\, \forall y\ q(x, z, y)$.

**6:** My good old friend John says that he has written a C++ program $P$ that takes as input an arbitrary first-order formula $F$, and such that, if $F$ is a tautology, $P$ always outputs "yes" after a finite amount of time, and if $F$ is not a tautology, $P$ outputs "no" or it does not terminate.

Is this possible? If this is not possible, explain why. If it is possible, explain how $P$ would work. **A very short answer suffices.**