LAB 3

S is a finite set of propositional clauses, written in CNF in the format [[w, s, n (p)], [a, n (w), r, t], [q]]. With n(p) the negation of p is noted.

1. Implement the Resolution procedure. For the input data S, the procedure will display SATISFIABLE, respectively UNSATISFIABLE, as it is the case. Set a strategy for choosing which pair of clauses to use when Resolution rule is applied.

2. Implement the Davis-Putnam SAT procedure. For the input data S, the procedure will display YES, respectively NOT, as S is satisfiable or not. In the case of YES, the procedure will also display the truth values assigned to the literals (i.e. the solution {w/true; s/false; p/false ...}). Choose two strategies of selection of the atom to perform the • operation and compare the results.

The input data will be read from a file and the results will be displayed in a file.

Both procedures will be implemented in the versions presented at the course (from Ronald Brachman, Hector Levesque. Knowledge representation and reasoning, Morgan Kaufmann 2004).

Each subject will receive a grade. For 10, the implementation will be done in PROLOG. If you use another programming language, the maximum mark will be 8.
!!! THE CODE WILL HAVE NO COMMENTS AT ALL. A program with any comments included will not be considered at all.

For each subject:
- 2p the language
- 1p reading/writing as required
- 3p the program running on different examples
- 4p explanations of the code that implements the problem

Reading/writing in Prolog

The data in the input file is separated by .

```
see (user).       % opens the current input environment - the user represents the keyboard
see ("c:\\prolog\\a.txt ').
seen.    % closes the current reading environment

read (X).
read (end_of_file).      % special atom that detects the end of the file
```

```
tell ("c:\\prolog\\a.txt ').          % opens the current output environment
told.     % closes the current output environment

write(parent(ion,maria)).
```

Dynamic predicates in Prolog

-declared with   :-dynamic p/1.   %PredicateName/arity

-predicates for dynamic addition: asserta, assertz, assert (add at the beginning, at the end)
-predicates for dynamic deletion: retract, retractall.

```
:-dynamic fib/2.

fib(1,1).
fib(2,1).
fib(N,F):-N>2, N1 is N-1, fib(N1,F1), N2 is N-2, fib(N2,F2), F is F1+F2, asserta(fib(N,F):-!).
```