**Problem Statement:** Implement in LISP a theorem prover based on propositional resolution which implements the algorithm discussed below:

==================

Let F be a set of axioms and P the theorem to prove.
Convert all the propositions of F and ¬ P to clause form.
Repeat until either a contradiction is found or no progress can be made (i.e., no new clauses may be generated):

- Select two clauses.
- Resolve them together (by applying the resolution rule).
- If the resulting clause (the resolvent) is the empty clause (i.e., contains no element), then a contradiction has been found, or else add the resulting clause to the set of clauses available to the algorithm.

==================

__**Conventions required:**__

You will assume that the axioms and the negation of the theorem to prove are already in clause form.
Each clause will be represented as a list of two elements.
The first element is a list containing the propositions from the left-hand side of the clause.
The second element is a list containing the propositions from the right-hand side of the clause.

That is, the clause
    A1,..., Am ← B1,...,Bn,X
should be represented as
    ((A1 ... Am) (B1 ... Bn X))
In particular
    A1, A2 ←
should be represented as
    ((A1 A2) ())

The main function of your system shall be called "resolution," and shall have only one parameter, as shown in the following:

```
(defun resolution (lc)
;; lc is a list of at least two propositional clauses representing
;; A set of axioms and the negation of the theorem to prove.
```

The function "resolution" will print the message "A contradiction had been found" and will return NIL, or it will print the message "There is no contradiction in the given set of clauses" and will return T, as illustrated in the following examples:

```
→ (resolution '( ((C D)(A)) ((A D E)()) (()(A C)) (()(D)) (()(E)) ))
  A contradiction had been found
  NIL

→ (resolution '( ((A)(B C)) ((B)()) (()(A)) ((C)()) ))
  A contradiction had been found
  NIL

→ (resolution '( ((A)(B C)) ((B)()) ((C)()) ))
  There is no contradiction in the given set of clauses
  T
```