Implement a proof by resolution system for propositional logic.

This resolution technique uses proof by contradiction and is based on the fact that any sentence in propositional logic can be transformed into an equivalent sentence in conjunctive normal form.

The steps are as follows:

1. All sentences in the knowledge base and the negation of the sentence to be proved (the conjecture) are conjunctively connected.

2. The resulting sentence is transformed into a conjunctive normal form with the conjuncts viewed as elements in a set, S, of clauses. For example, (a_1 or a_2) and (neg a_2 or b_2 or b_3) and (neg b_2 or b_3) and (neg b_3) and (neg a_1) gives rise to the set S = {a_1 or a_2, neg a_2 or b_2 or b_3, neg b_2 or b_3, neg b_3, neg a_1}.

3. The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the sentence contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set S, it is added to S, and is considered for further resolution inferences.

4. If after applying a resolution rule the empty clause is derived, the original formula is unsatisfiable (or contradictory), and hence it can be concluded that the initial conjecture follows from the axioms.

5. If, on the other hand, the empty clause cannot be derived, and the resolution rule cannot be applied to derive any more new clauses, the conjecture is not a theorem of the original knowledge base.

Your program should read disjunctions (one disjunction per line from a file) using the Prolog see and read predicates, where the last row is the query to the program. Example input:

```
myClause(1, or(a_1, a_2)).

myClause(2, or(or(neg(a_2), b_2), b_3)).
myClause(3, or(neg(b_2), b_3)).
myClause(4, neg(b3)).
myQuery(5, a_1).
```

Your program should write prolog facts to an output file containing the graph representation of which clauses were resolved to derive each resolvent and if the empty clause was derived or not.

Example output:

```
success.

resolution(1, 5, a_2, 6).

resolution(6, 2, or(b_2, b_3), 7).

resolution(7, 3, b_3, 8).

resolution(8, 4, empty, 9).
```