```
# Example propositional logic statements in CNF
# These are the given statements after conversion to CNF
kb = [
    {"¬B", "¬C", "A"},  # ¬B ∨ ¬C ∨ A
    {"B"},  # B
    {"¬D", "¬E", "C"},  # ¬D ∨ ¬E ∨ C
    {"E", "F"},  # E ∨ F
    {"D"}, #D
     {"¬F"}, #¬F
]

# Negate the query: If the query is "A", we negate it to "¬A"
def negate_query(query):
    if "¬" in query:
        return query.replace("¬", "")  # If it's negated, remove the negation
    else:
        return f"¬{query}"  # Otherwise, add negation in front

# Function to perform resolution on two clauses
def resolve(clause1, clause2):
    resolved_clauses = []

    # Try to find complementary literals
    for literal1 in clause1:
        for literal2 in clause2:
            # If literals are complementary (e.g., "A" and "¬A"), resolve them
            if literal1 == f"¬{literal2}" or f"¬{literal1}" == literal2:
                new_clause = (clause1 | clause2) - {literal1, literal2}
                resolved_clauses.append(new_clause)

    return resolved_clauses

# Perform resolution-based proof
def resolution(kb, query):
    # Step 1: Negate the query and add it to the knowledge base
    negated_query = negate_query(query)
    kb.append({negated_query})

    # Step 2: Initialize the set of clauses
    new_clauses = set(frozenset(clause) for clause in kb)

    while True:
        resolved_this_round = set()
        clauses_list = list(new_clauses)

        # Try to resolve every pair of clauses
        for i in range(len(clauses_list)):
            for j in range(i + 1, len(clauses_list)):
                clause1 = clauses_list[i]
                clause2 = clauses_list[j]

                # Apply resolution to the two clauses
                resolved = resolve(clause1, clause2)
                if frozenset() in resolved:
                    return True  # Found an empty clause (contradiction), query is provable
                resolved_this_round.update(resolved)

        # If no new clauses were added, stop
        if resolved_this_round.issubset(new_clauses):
            return False  # No new clauses, query is not provable

        # Add new resolved clauses to the set
        new_clauses.update(resolved_this_round)
```

```
# Query to prove: "A"
query = (input("Enter the query:"))
result = resolution(kb, query)
print("OUTPUT:(1BM22CS300)")
print("Using Resolution to prove a query")
# Print the result
print(f"Is the query '{query}' provable? {'Yes' if result else 'No'}")
```

```
Enter the query:A
OUTPUT:(1BM22CS300)
Using Resolution to prove a query
Is the query 'A' provable? Yes
```