```python
class CaseBasedReasoning:
    def __init__(self):
        self.case_base = []  # List to store past cases (problem, solution) pairs

    def add_case(self, problem, solution):
        """
        Adds a new case to the case base.
        """
        self.case_base.append((problem, solution))

    def retrieve_similar_cases(self, new_problem):
        """
        Finds the most similar cases from the case base based on a simple similarity metric.
        """
        best_match = None
        max_similarity = 0
        for case in self.case_base:
            similarity = self.calculate_similarity(new_problem, case[0])  # Calculate similarity
            if similarity > max_similarity:
                max_similarity = similarity
                best_match = case
        return best_match

    def calculate_similarity(self, problem1, problem2):
        """
        A basic similarity calculation (can be customized based on the problem domain).
        """
        # Example: Comparing key attributes of problems
        shared_attributes = sum(1 for attr in problem1 if attr in problem2)
        return shared_attributes / len(problem1)

    def predict_solution(self, new_problem):
        """
        Predicts a solution for a new problem by retrieving the most similar case.
        """
        similar_case = self.retrieve_similar_cases(new_problem)
        if similar_case:
            return similar_case[1]  # Return the solution from the similar case
        else:
            return None  # No similar case found
cbr = CaseBasedReasoning()

# Add some initial cases
cbr.add_case(["fever", "headache", "tired"], "flu")
cbr.add_case(["sore throat", "cough", "runny nose"], "common cold")

# Predict solution for a new case
new_problem = ["fever", "cough", "muscle aches"]
predicted_solution = cbr.predict_solution(new_problem)
print(predicted_solution)  # Output: "flu" (assuming the similarity calculation favors "fever"
and "muscle aches")
```