Lab 2

Sets + Inference Rules

1 Part 1: Power Set

Given a set represented as array list of distinct strings, you have to generate the power set from the set. You have to implement the requirement in two ways:

1) Recursive approach
2) Iterative approach

You are **not allowed** to use built in Set data structure or any data structure similar to it.

2 Part 2: Logical Expressions Solver

Design and implement a robust parser capable of handling logical expressions. The parser should support a simplified syntax for propositional logic, including the parentheses for grouping and the following operations

- negation (~)
- conjunction (^)
- disjunction (v)
- implication (>)

Your program should take, as input, the expression and prompt the user for the boolean values of each variable in the expression. It should then evaluate the expression based on the given boolean values and output the result. Make sure to follow the precedence rules of the operators that were discussed in class.

Example 1:

Input

- expression is "(P ^ Q) v ~R"
- P = true, Q = false and R = true

Output

- false

Example 2:

Input

- expression is "(P ^ Q) > ~R"
- P = true, Q = false and R = true

Output

- true

Example 3:

Input

- expression is "(P ^ Q) > ^ ~R"

Output

- Wrong expression

You are required use following interfaces:

```
interface Expression {
  String getRepresentation();
  void setRepresentation(String representation);
}
interface LogicalExpressionSolver {
  boolean evaluateExpression(Expression expression);
}
```

3 Part 3: Inference Engine

Design and implement inference engine that can apply inference rules to logical expressions. The engine should take logical expressions as input, identify applicable inference rules, and generate the corresponding output based on the rules' logic. The supported inference rules are as follows:

- Modus ponens. Given expressions of the form "P > Q" and "P", the rule allows inferring "Q"
- Modus tollens. Given expressions of the form "P > Q" and "~Q", the rule allows inferring "~P"
- Hypothetical syllogism. Given expressions of the form "P > Q" and "Q > R", the rule allows inferring "P > R"
- Disjunctive syllogism. Given expressions of the form "P v Q" and "~P", the rule allows inferring "Q"
- Resolution. Given expressions of the form "P v Q" and "~P v R", the rule allows inferring "Q v R"

Your program should take two expressions as input and output the result of the inference process along with the applied rule. Assume that the inputs are simple expressions without parentheses and that each expression has at most one binary operation.

Example 1:

Input

- expressions are "~X v Y" and "X v Z"

Output

- Y v Z (Resolution)

Example 2:

Input

- expressions are "a > b" and "a > r"

Output

- The input expression cannot be inferred

You are required to use following interfaces:

```
interface InferenceRule {
  boolean matches(Expression exp1, Expression exp2);
  Expression apply(Expression exp1, Expression exp2);
}
interface InferenceEngine {
  void addRule(InferenceRule rule);
  void addExpression(Expression exp);
  Expression applyRules();
}
```

4 Submission

- You must work **in groups of two** and use **Java programming language** in your implementation.
- You should deliver all the coding files.
- Make sure you provide a clear and detailed report. It should contain:
  1. Problem statement.
  2. Used data structures.
  3. Sample runs and different test cases.
  4. Assumptions and details you find necessary to be clarified.

Good Luck,,,