```
from sympy.logic.boolalg import Or, And, Not, Implies
from sympy.logic import simplify_logic
from sympy import symbols

# Function to create a Knowledge Base by taking user input
def create_knowledge_base():
    # User input for number of propositions
    num_propositions = int(input("Enter the number of propositions: "))

    # Create symbols for the propositions
    propositions = [symbols(input(f"Enter the name of proposition {i+1}: ")) for i in range(num_propositions)]

    kb = []

    # Allow the user to input knowledge base rules (as logical expressions)
    print("Enter the knowledge base as one logical expression, using 'and', 'or', 'not', 'implies' (use parentheses for grouping):")
    print("Example: (Q implies P) and (P implies not Q) and (Q or R)")
    rule = input("KB: ")

    try:
        # Parse the user input into sympy logical expressions
        rule_expr = parse_logical_expression(rule, propositions)
        kb.append(rule_expr)
    except Exception as e:
        print(f"Error parsing KB: {e}")

    return kb, propositions

# Function to parse a logical expression entered by the user
def parse_logical_expression(expr, propositions):
    # Replace user input keywords with sympy equivalents
    expr = expr.lower()
    expr = expr.replace("and", "&").replace("or", "|").replace("not", "~").replace("implies", ">>")

    # Create a dictionary of symbols
    symbol_dict = {prop.name: prop for prop in propositions}

    # Replace the proposition names with sympy symbols in the expression
    for prop in symbol_dict:
        expr = expr.replace(prop, f"symbols('{prop}')")

    # Use sympy.logic functions to parse and return the logical expression
    from sympy import sympify
    try:
        return sympify(expr)
    except Exception as e:
        raise ValueError(f"Failed to parse expression: {expr}. Error: {e}")

# Function to check if a query is entailed by the knowledge base
def entails(kb, query, symbols_list):
    # Simplify the knowledge base by combining all formulas into a single expression
    combined_kb = And(*kb)

    # Check if the query is a logical consequence of the knowledge base
    # If combined KB ∧ ¬query is unsatisfiable, then KB entails query
    result = simplify_logic(And(combined_kb, Not(query)))

    # If the result is False (unsatisfiable), the query is entailed
    if result == False:
        return True
    else:
        return False

# Example usage:
```

```
# Create knowledge base
kb, symbols_list = create_knowledge_base()

# Allow the user to input a query
query_str = input("Enter the query to check for entailment (use 'and', 'or', 'not', 'implies'): ")

# Parse the query entered by the user
try:
    query = parse_logical_expression(query_str, symbols_list)

    # Check if the query is entailed by the KB
    if entails(kb, query, symbols_list):
        print(f"The query {query_str} is entailed by the knowledge base.")
    else:
        print(f"The query {query_str} is not entailed by the knowledge base.")
except Exception as e:
        print(f"Error: {e}")
```

```
Enter the number of propositions: 3
Enter the name of proposition 1: P
Enter the name of proposition 2: Q
Enter the name of proposition 3: R
Enter the knowledge base as one logical expression, using 'and', 'or', 'not', 'implies' (use parentheses for grouping):
Example: (Q implies P) and (P implies not Q) and (Q or R)
KB: (Q implies P ) and (P implies not Q) and (Q or R)
Enter the query to check for entailment (use 'and', 'or', 'not', 'implies'): r
The query r is entailed by the knowledge base.
```