AI LAB EXAM – entailment algorithm

```
import itertools


def evaluate_expression(expression, model):

    try:
        for var, value in model.items():
            expression = expression.replace(var, str(value))
        return eval(expression)
    except Exception as e:
        print(f"Error evaluating expression: {e}")
        return False

def get_propositional_symbols(formula):

    symbols = set()
    for char in formula:
        if 'A' <= char <= 'Z':
            symbols.add(char)
    return sorted(list(symbols))

def entails(knowledge_base, query):
    all_symbols = sorted(list(get_propositional_symbols(knowledge_base)
or get_propositional_symbols(query)))

    for assignment_values in itertools.product([True, False],
repeat=len(all_symbols)):
        model = dict(zip(all_symbols, assignment_values))

        kb_true_in_model = evaluate_expression(knowledge_base, model)

        if kb_true_in_model:
            query_true_in_model = evaluate_expression(query, model)
            if not query_true_in_model:
                return False
    return True

if __name__ == "__main__":
    print("Connectors: 'and', 'or', 'not', uppercase letters for
variables.")

    knowledge_base_input = input("Knowledge Base: ")
    query_input = input("Query: ")

    if entails(knowledge_base_input, query_input):
        print(f"Knowledge Base '{knowledge_base_input}' entails the
Query '{query_input}'.")
```

```
    else:
        print(f"Knowledge Base '{knowledge_base_input}' doesnt entail
the Query '{query_input}'.")
```

```
            print(f"Knowledge Base '{knowledge_base_input}' doesnt entail the Query '{que
```

```
Connectors: 'and', 'or', 'not', uppercase letters for variables.
Knowledge Base: (A or C) and (B or not C)
Query: A or B
Knowledge Base '(A or C) and (B or not C)' entails the Query 'A or B'.
```