USN:1BM22CS259

LAB-6: Propositional Logic

CODE:

```
import itertools

# Function to evaluate if a sentence is true in the given model
def pl_true(sentence, model):
    # Extract truth values for the variables from the model
    A = model.get('A', False)
    B = model.get('B', False)
    C = model.get('C', False)

    if sentence == "A or B":
        return A or B
    elif sentence == "(A or C) and (B or not C)":
        return (A or C) and (B or not C)
    return False

# TT-ENTAILS? function: returns true if KB entails alpha
def tt_entails(kb, alpha):
    symbols = ['A', 'B', 'C']  # List of all propositional symbols
    return tt_check_all(kb, alpha, symbols, {})

# TT-CHECK-ALL function: recursively checks all possible models
def tt_check_all(kb, alpha, symbols, model):
    if not symbols:  # If there are no more symbols to assign
        if pl_true(kb, model):
            return pl_true(alpha, model)  # Return true if both KB and α are true in the model
        else:
```

```
            return True  # If KB is false, return true (trivially satisfied)
    else:
        p = symbols[0]  # Get the first symbol
        rest = symbols[1:]  # Remaining symbols

        # Create two new models: one where p is true and one where p is false
        model_true = model.copy()
        model_false = model.copy()
        model_true[p] = True
        model_false[p] = False

        # Recursively check both models
        return (tt_check_all(kb, alpha, rest, model_true) and
                tt_check_all(kb, alpha, rest, model_false))

# Knowledge base and alpha (proposition) in string format
kb = "(A or C) and (B or not C)"
alpha = "A or B"

# Check if KB entails alpha
result = tt_entails(kb, alpha)
print(f"KB entails α: {result}\n")

# Function to generate and print both the full truth table and the entailment table
def generate_truth_tables():
    print("Full Truth Table:")
    print(f"{'A':<10}{'B':<10}{'C':<10}{'A∨C':<10}{'B∨¬C':<10}{'KB':<10}{'α (A∨B)':<10}")
    full_table = []
```

```
    for A, B, C in itertools.product([False, True], repeat=3):
        A_or_C = A or C
        B_or_not_C = B or not C
        KB = (A or C) and (B or not C)
        alpha = A or B
        full_table.append((A, B, C, A_or_C, B_or_not_C, KB, alpha))
        print(f"{str(A):<10}{str(B):<10}{str(C):<10}{str(A_or_C):<10}{str(B_or_not_C):<10}{str(KB):<10}{str(alpha):<10}")

    print("\nEntailment Table (Only rows where KB and α are true):")
    print(f"{'A':<10}{'B':<10}{'C':<10}{'A∨C':<10}{'B∨¬C':<10}{'KB':<10}{'α (A∨B)':<10}")

    for row in full_table:
        A, B, C, A_or_C, B_or_not_C, KB, alpha = row
        if KB and alpha:
            print(f"{str(A):<10}{str(B):<10}{str(C):<10}{str(A_or_C):<10}{str(B_or_not_C):<10}{str(KB):<10}{str(alpha):<10}")

generate_truth_tables()
```

OUTPUT:

```
KB entails α: True

Full Truth Table:
A         B         C         A∨C       B∨¬C      KB        α (A∨B)
False     False     False     False     True      False     False
False     False     True      True      False     False     False
False     True      False     False     True      False     True
False     True      True      True      True      True      True
True      False     False     True      True      True      True
True      False     True      True      False     False     True
True      True      False     True      True      True      True
True      True      True      True      True      True      True

Entailment Table (Only rows where KB and α are true):
A         B         C         A∨C       B∨¬C      KB        α (A∨B)
False     True      True      True      True      True      True
True      False     False     True      True      True      True
True      True      False     True      True      True      True
True      True      True      True      True      True      True
```