

LAB-6: Propositional

Logic

CODE:

```
import itertools
```

```
# FuncΘon 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? funcΘon: returns true if KB entails alpha
```

```
def  $\Sigma$ _entails(kb, alpha):
```

```
    symbols = ['A', 'B', 'C'] # List of all proposiΘonal symbols
```

```
    return  $\Sigma$ _check_all(kb, alpha, symbols, {})
```

```
# TT-CHECK-ALL funcΘon: recursively checks all possible models
```

```
def  $\Sigma$ _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  $\alpha$  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 ( $\Sigma\_check\_all(kb, alpha, rest, model\_true)$  and
                 $\Sigma\_check\_all(kb, alpha, rest, model\_false)$ )

# 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 =  $\Sigma\_entails(kb, alpha)$ 
print(f"KB entails  $\alpha$ : {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  $\vee$  C':<10}{'B  $\vee$   $\neg$  C':<10}{'KB':<10}{' $\alpha$  (A  $\vee$  B)':<10}")
    full_table = []

```

