

# Sanghamitra R

## LAB 7

### Propositional Logic

#### Code:

```
from itertools import product

def pl_true(sentence, model):
    """Evaluates if a sentence is true in a given model."""
    if isinstance(sentence, str):
        return model.get(sentence, False)
    elif isinstance(sentence, tuple) and len(sentence) == 2: # NOT operation
        operator, operand = sentence
        if operator == "NOT":
            return not pl_true(operand, model)
    elif isinstance(sentence, tuple) and len(sentence) == 3:
        operator, left, right = sentence
        if operator == "AND":
            return pl_true(left, model) and pl_true(right, model)
        elif operator == "OR":
            return pl_true(left, model) or pl_true(right, model)
        elif operator == "IMPLIES":
            return not pl_true(left, model) or pl_true(right, model)
        elif operator == "IFF":
            return pl_true(left, model) == pl_true(right, model)

def print_truth_table(kb, query, symbols):
    """Generates and prints the truth table for KB and Query."""
    # Define headers with spaces for alignment
    headers = ["A", "B", "C", "A V C", "B V ¬C", "KB", "α"]
    print(" | ".join(headers))
    print("-" * (len(headers) * 9)) # Separator line

    # Generate all combinations of truth values
    for values in product([False, True], repeat=len(symbols)):
        model = dict(zip(symbols, values))

        # Evaluate sub-expressions and main expressions
        a_or_c = pl_true(("OR", "A", "C"), model)
        b_or_not_c = pl_true(("OR", "B", ("NOT", "C")), model)
        kb_value = pl_true(("AND", ("OR", "A", "C"), ("OR", "B", ("NOT", "C"))), model)
        alpha_value = pl_true(("OR", "A", "B"), model)

        # Print the truth table row
        row = values + (a_or_c, b_or_not_c, kb_value, alpha_value)
        row_str = " | ".join(str(v).ljust(7) for v in row)
```

```

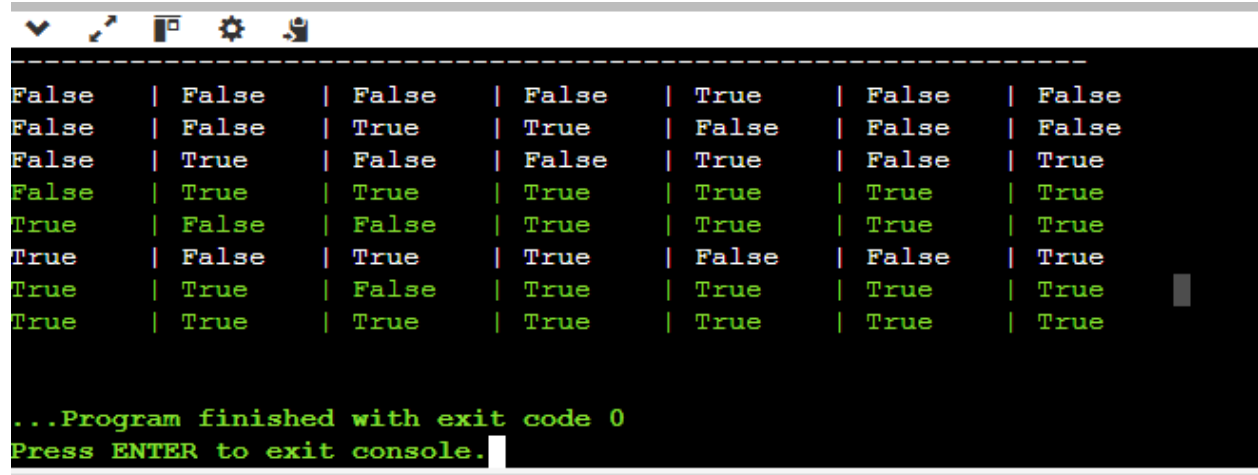
# Highlight rows where both KB and  $\alpha$  are true
if kb_value and alpha_value:
    print(f"\033[92m{row_str}\033[0m") # Green color for rows where KB and  $\alpha$  are true
else:
    print(row_str)

# Define the knowledge base and query
symbols = ["A", "B", "C"]
kb = ("AND", ("OR", "A", "C"), ("OR", "B", ("NOT", "C")))
query = ("OR", "A", "B")

# Print the truth table
print_truth_table(kb, query, symbols)

```

### Output:



```

-----
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

...Program finished with exit code 0
Press ENTER to exit console.

```