6.PROPOSITIONAL ENTAILMENT BY IMPLEMENTING TRUTH TABLE ENUMERATION

```
def PL_TRUE(model, sentence):
    """
    Evaluates the sentence in the context of the given model (truth
assignment).
    The sentence is a logical expression involving variables A, B, C.
    """
    # Replace variables with their truth values in the model
    sentence = sentence.replace("A", str(model["A"])).replace("B",
str(model["B"])).replace("C", str(model["C"]))

    # Replace logical operations with Python's logical operations
    sentence = sentence.replace("and", "and").replace("or",
"or").replace("not", "not")

    # Evaluate the sentence in Python's eval function
    return eval(sentence)

def truth_table_entailment(KB, query, variables):
    """
    This function checks if the KB entails the query using truth table
enumeration.
    It evaluates all possible truth assignments for the given variables.
    """
    all_assignments = []

    # Generate all possible truth assignments for the variables
    for i in range(2 ** len(variables)):
        assignment = {}
        for j, var in enumerate(variables):
            assignment[var] = (i >> j) & 1  # 0 or 1 assignment for each
variable
        all_assignments.append(assignment)

    # Check for each truth assignment
    for assignment in all_assignments:
        # Check if the KB is true for this assignment
        kb_true = PL_TRUE(assignment, KB)
        query_true = PL_TRUE(assignment, query)
        # If KB is true and Query is false in any row, return False
        if kb_true and not query_true:
            return False

    return True

# Example usage

# Define KB and Query in propositional logic
KB = "(A or C) and (B or not C)"  # Knowledge Base
query = "A or B"  # Query

# Variables involved in the KB and Query
variables = ['A', 'B', 'C']

# Call the function to check if KB entails the Query
result = truth_table_entailment(KB, query, variables)

# Output the result
print("Does KB entail the query?", result)
```

OUTPUT:

```
Does KB entail the query? True
```

12/11/24
Tuesday

Lab-6

Propositional Entailment by implementation of Truth-Table Enumeration

Algorithm:

```
def PL_TRUE (model, Sentence):
  Sentence = Sentence. replace ("A", Str (model ["A"])).
replace ("B", Str (model ["B"])). replace
    ("C, Str (model ["C"]))
Sentence = sentence. replace ("and", "and").
replace ("or", "or"). replace ("not", "not")
  return eval (Sentence)

def truth_table_Entalment (KB, query, Variables)
 all_assignments = [ ]
for i in range (2** len (variables)):
  assignment = { }
for j, var in Enumerate (variables):
  assignment [var] = (i >>j) & 1
  all_assignments.append (assignment)

for assignment in all_assignments:
Kb_true = PL_TRUE (assignment, KB)
 query_true = PL_TRUE (assignment, query)
 if KB_true and not query_true:
      return False
return True
KB = "(A or C) and (B or not C)"
query = "A or B"
 variables = ['A', 'B', 'C']
result = truth_table_Entaiment (KB, query, Variables)
print ("Does KB Entail the query ?", result)
```

Output:
Does KB Entail the query? True.

Truth Table:

A	B	C	AUC	BU¬C	KB=(AUC)∩(BU¬C)	α = AUB
T	T	T	T	T	T	T
T	T	F	T	T	T	T
T	F	T	T	F	F	T
T	F	F	T	T	T	T
F	T	T	T	T	T	T
F	T	F	F	T	F	T
F	F	T	T	F	F	F
F	F	F	F	T	F	F