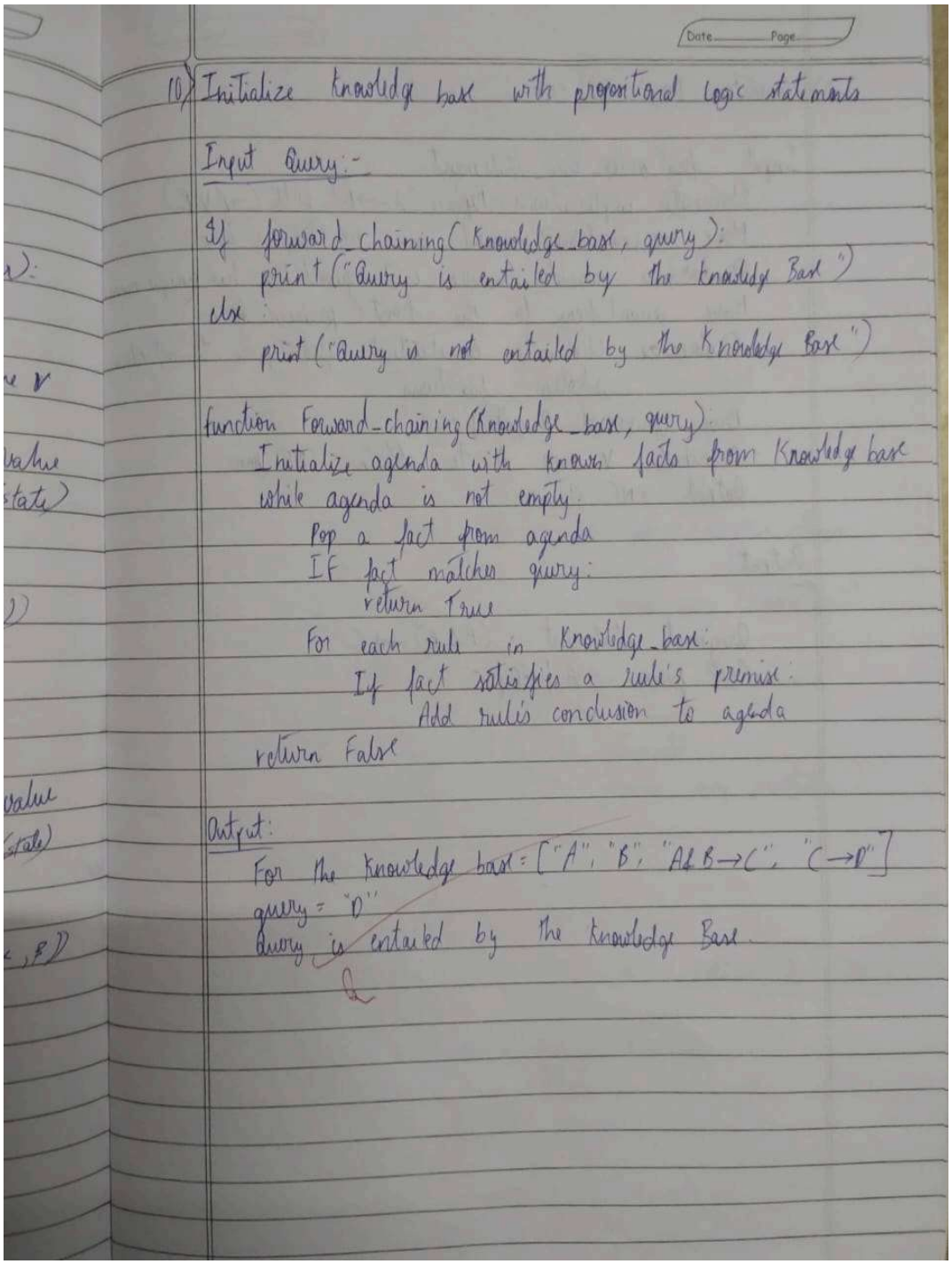
Entailment Algorithm:



10) Initialize knowledge base with propositional logic statements

Input Query:-

```
if forward_chaining(Knowledge_base, query):
    print("Query is entailed by the Knowledge Base")
else:
    print("Query is not entailed by the Knowledge Base")

function Forward-chaining(Knowledge_base, query):
    Initialize agenda with known facts from Knowledge base
    while agenda is not empty:
        Pop a fact from agenda
        If fact matches query:
            return True
        For each rule in Knowledge_base:
            If fact satisfies a rule's premise:
                Add rule's conclusion to agenda
    return False
```

Output:

For the knowledge base = ["A", "B", "A & B → C", "C → D"]
query = "D"
Query is entailed by the Knowledge Base

Code:

```
from itertools import product


# Define a function to evaluate a propositional expression
def evaluate(expr, model):
    """
    Evaluates the given expression based on the values in the model.
    """
    for var, val in model.items():
        expr = expr.replace(var, str(val))
    return eval(expr)


# Define the truth-table enumeration algorithm
def truth_table_entails(KB, query, symbols):
    """
    Checks if KB entails query using truth-table enumeration.
    KB: list of propositional expressions (strings)
    query: propositional expression (string)
    symbols: list of symbols (propositions) in the KB and query
    """
    # Generate all possible truth assignments
    assignments = list(product([False, True], repeat=len(symbols)))


    entailing_models = []


    # Iterate over each assignment to check entailment
    for assignment in assignments:
        model = dict(zip(symbols, assignment))


        # Check if KB is true in this model
```

```
        KB_is_true = all(evaluate(expr, model) for expr in KB)

        # If KB is true, check if query is also true
        if KB_is_true:
            query_is_true = evaluate(query, model)
            if query_is_true:
                entailing_models.append(model)  # Store the model
            else:
                return False, []
               # Found a model where KB is true but query is false

    return True, entailing_models  # KB entails query if no counterexample was
found

# Get input from the user
symbols = input("Enter the propositions (symbols) separated by spaces:
").split()
KB = []
n = int(input("Enter the number of statements in the knowledge base: "))

for i in range(n):
    expr = input(f"Enter statement {i + 1} in the knowledge base: ")
    KB.append(expr)

query = input("Enter the query: ")

# Check entailment
result, models = truth_table_entails(KB, query, symbols)
if truth_table_entails(KB, query, symbols):
    print("KB entails the query.")
    print("Models where KB entails query:")
```

```
    for model in models:
        print(model)
else:
    print("KB does not entail the query.")
```

Output:

Enter the propositions (symbols) separated by spaces: A B C

Enter the number of statements in the knowledge base: 2

Enter statement 1 in the knowledge base: A or C

Enter statement 2 in the knowledge base: B or not C

Enter the query: A or B

KB entails the query.

Models where KB entails query:

{'A': False, 'B': True, 'C': True}

{'A': True, 'B': False, 'C': False}

{'A': True, 'B': True, 'C': False}

{'A': True, 'B': True, 'C': True}