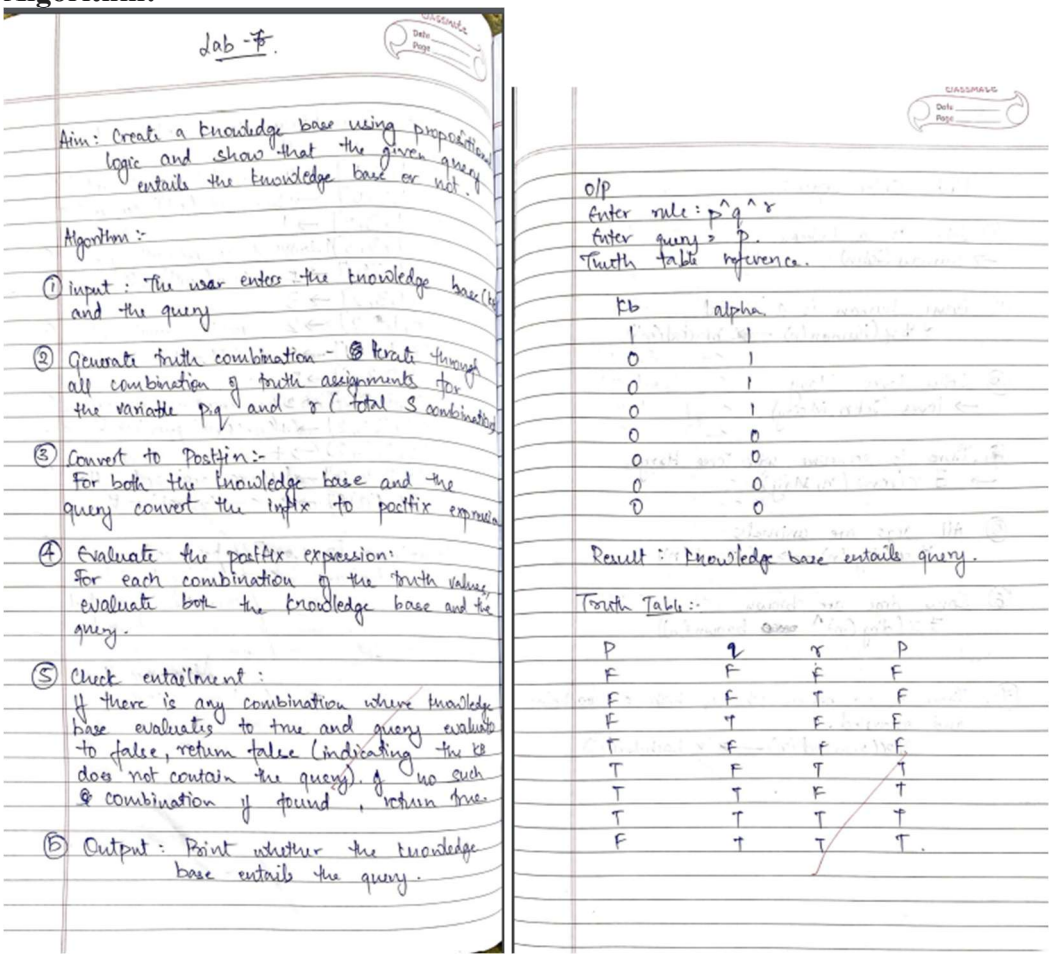
Program6:

Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not.

Algorithm:



Lab - 6.

Aim: Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not.

Algorithm:

1. Input: The user enters the knowledge base (KB) and the query
2. Generate truth combination - Iterate through all combination of truth assignments for the variable p, q and r (total 8 combinations)
3. Convert to Postfix:- For both the knowledge base and the query convert the infix to postfix expression
4. Evaluate the postfix expression: For each combination of the truth values, evaluate both the knowledge base and the query.
5. Check entailment: If there is any combination where knowledge base evaluates to true and query evaluates to false, return false (indicating the KB does not contain the query). If no such combination if found, return true.
6. Output: Print whether the knowledge base entails the query.

O/P

Enter rule: p^q^r

Enter query: p

Truth table reference.

KB	alpha
1	1
0	1
0	1
0	1
0	0
0	0
0	0
0	0

Result: Knowledge base entails query.

Truth Table:

p	q	r	p
F	F	F	F
F	F	T	F
F	T	F	F
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T
F	T	T	T

Code:

```
def implies(p, q):
    return not p or q

def print_truth_table(KB, alpha, sym):
    print("Truth Table:")
    header = " | ".join(sym) + " | KB | alpha "
    print(header)
    print("-" * len(header))

    for values in product([True, False], repeat=len(sym)):
        model = dict(zip(sym, values))
        kb_true = all(statement(model) for statement in KB)
        query_true = query(model)
```

```
        row = " | ".join(str(val) for val in values) + f" | {kb_true} | {query_true}"
        print(row)

def check_entailment(KB, query, sym):

    for values in product([True, False], repeat=len(sym)):
        model = dict(zip(sym, values))
        kb_true = all(statement(model) for statement in KB)
        query_true = query(model)
        if kb_true and not query_true:
            return False
    return True
sym=['P', 'Q', 'R']
KB = [
    lambda model: implies(model['Q'], model['P']),
    lambda model: implies(model['P'], not model['Q']),
    lambda model: model['Q'] or model['R']
]
alpha = lambda model: model['R']


entails = check_entailment(KB, alpha,sym)
print(f"KB entails alpha: {entails}")
print_truth_table(KB, alpha,sym)
```

Output Snapshot:

```
KB entails alpha: True
Truth Table:
P | Q | R | KB | alpha
----------------------
True | True | True | False | True
True | True | False | False | False
True | False | True | True | True
True | False | False | False | False
False | True | True | False | True
False | True | False | False | False
False | False | True | True | True
False | False | False | False | False
```