Program6:

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

Algorithm:

dab - F. Page	
Aim: Create a tracidedge base using Proposition logic and show that the given green green entails the tracidedge base or not.	CIASSANAG Doly Rege
entails the tripictedge base or not of	OIP
	futer rule: p^q 2
Higoritim:	Aver Auni 2 D
A Maria Commission of the Maria Commission of the Commission of th	Thath table reverse.
1) input: The war enter the thousedge base (is	(MILITA PORT)
mput: (the same that	to lathe a mount man
and the query	Policinal A - almost select
Querate inthe combination - @ terate through	0 3>
general min combination	7.0
all combination of truth assignments for the variable pig and of total sandington	0 1 / Minute and c
the vanatu pig and of the sambination	0 1 5 1 (25 hi whi 140) C
	Orals and One man of man, in
Convert to Postfin:	Orrest and Orrest and area of
For both the knowledge base and the	O O O ON W COME &
query convert the infix to pacifix express	0 0
Dan to be designed to	South an sol It 3
F Evaluate the postfix expression:	Result: Montedge base estails givery.
For each combination of the touth value evaluate both the knowledge base and the	Trou
	Touth Table: Sund asso Alan while the
grey.	
	P P F F F
Check entailment:	6 6 7 6
If there is any combination where mailedge	E T F
have evalualis to true and guery evaluate	Ottom of the state
the there is any combination where thankly base evaluates to true and grany evaluates to false, return tale (indicating the is does not contain the query). I no such a combination of found, return true	T F 7 1
as a not contain the query). I no even	T 7 F / T
tombination of found, repair me	7 7 7 7
	F T T
6 Output: Point whether the triondedge	, , , , , , , , , , , , , , , , , , , ,
base entails the query.	
1	,

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
```