ICE-7

Ananya Samala- 11527196

1. Negation: ¬p

In [22]:

```
def negation (p):
  return not p

print ("p\t~p")

for p in [True, False]:
  a= negation (p)
  print (p,a)

```

```
p       ~p
True False
False True
```

2.Conjunction: p ∧ q

In [67]:

```
def conjunction (p,q):
  return p and q

print ('P\tQ P^Q')

for p in [True, False]:
  for q in[True, False]:
    a= conjunction(p,q)
    print (p, q, a)
```

```
P       Q P^Q
True True True
True False False
False True False
False False False
```

3.Disjunction: p ∨ q

In [35]:

```python
def disjunction (p,q):
  return p or q

print ('P\tQ p ∨ q')

for p in [True, False]:
  for q in[True, False]:
    a= disjunction(p,q)
    print(p,q,a)


```

```
P       Q p ∨ q
True True True
True False True
False True True
False False False
```

4.Implication: p ⟹ q

In [36]:

```python
def implication (p,q):
  return not (p) or q

print ('P\tQ p ⟹ q')

for p in [True, False]:
  for q in[True, False]:
    a= implication (p,q)
    print(p,q,a)


```

```
P       Q p ⟹ q
True True True
True False False
False True True
False False True
```

5.Bi-Implication: p ⟺ q

In [38]:

```python
def biimplication (p,q):
  return (a and p) or (not a and not p)

print ('P \tQ p ⟺ q')

for p in [True, False]:
  for q in[True, False]:
    a= biimplication (p,q)
    print(p,q,a)


```

```
P       Q p ⟺ q
True True True
True False True
False True False
False False True
```

6.Contradiction: p∧¬p

In [39]:

```python
def contradiction (p):
  return p and not(p)

def negation (p):
  return not p

print ('P \tQ p∧¬p')

for p in [True, False]:
  a= contradiction (p)
  b= negation (p)
  print (p,b,a)

```

```
P       Q p∧¬p
True False False
False True False
```

7.Compound Propositions

a. (p ∧ q) ∨ ¬q

In [40]:

```
print ('P \tQ p ∧ q) ∨ ¬q')

for p in [True, False]:
  for q in[True, False]:
    a= disjunction( conjunction(p,q), not(q))
    print (p,q,a)
```

```
P       Q p ∧ q) ∨ ¬q
True True True
True False True
False True False
False False True
```

b.(p∨¬q) ∧ ¬p

In [41]:

```
print ('P\tQ (p∨¬q) ∧ ¬p')

for p in [True, False]:
  for q in[True, False]:
    a= conjunction( disjunction(p, not(q)), not(p))
    print (p,q,a)
```

```
P       Q (p∨¬q) ∧ ¬p
True True False
True False False
False True False
False False True
```

c. (p ∧ q) → (p ∧ r) for all values of p, q, r

In [80]:

```python
def implies(p, q):
  if p == True and q == False:
    return False
  return True

equivalent = True
print("p q r (p ∧ q) → (p ∧ r)")

for p in range(2):
  for q in range(2):
    for r in range(2):
     if implies(conjunction(p, q),conjunction(p, r)) :
       equivalent = False
     print(p, q,r, "\t", implies(conjunction(p, q),conjunction(p, r)),

if equivalent:
  print()

else:
  print()
```

```
p q r (p ∧ q) → (p ∧ r)
0 0 0    True
0 0 1    True
0 1 0    True
0 1 1    True
1 0 0    True
1 0 1    True
1 1 0    False
1 1 1    True
```

8: Equivalent/not equivalent

a. (p ∧ q) → r and P → (q ∧ r)

In [58]:

```
def implies(p, q):
  if p == True and q == False:
    return 0
  return 1

equivalent = True
print("p q r\t(p ∧ q) → r \t P → (q ∧ r)")
for p in range(2):
  for q in range(2):
    for r in range(2):
      if implies(p and q, r) != implies(p, q and r):
        equivalent = False
      print(p, q, r, "\t", implies(p and q, r), "\t\t", implies(p, q an


if equivalent:
  print("\nexpressions are eqivalent")
else:
  print("\nexpressions not are eqivalent")

```

```
p q r   (p ∧ q) → r      P → (q ∧ r)
0  0  0     1               1
0  0  1     1               1
0  1  0     1               1
0  1  1     1               1
1  0  0     1               0
1  0  1     1               0
1  1  0     0               0
1  1  1     1               1

expressions not are eqivalent
```

b.$(p \longrightarrow q) \wedge (q \longrightarrow p)$ and $p \longleftrightarrow q$

In [50]:

```python
def implies(p, q):
  if p == True and q == False:
    return 0
  return 1

equivalent = True
print("p q (p → q)^(q → p) \tp ↔ q")
#print("p q (p\u2192q)\u2227(q\u2192p) p\u27F7q")

for p in range(2):
  for q in range(2):
    if implies(p, q) and implies(q, p) != int(p == q):
      equivalent = False
    print(p, q, "\t", implies(p, q) and implies(q, p), "\t" "\t ", int(

if equivalent:
  print("\nexpressions are eqivalent")

else:
  print("\nexpressions not are eqivalent")

```

```
p q (p → q)^(q → p)     p ↔ q
0 0      1                1
0 1      0                0
1 0      0                0
1 1      1                1

expressions are eqivalent
```

c.¬ (p → q) and p ∧ ¬q

In [53]:

```python
def implies(p, q):
  if p == True and q == False:
    return 0
  return 1

equivalent = True
print("p q   ¬ (p → q)   p ∧ ¬q")

for p in range(2):
  for q in range(2):
    if implies(p, q) and implies(q, p) != int(p == q):
      equivalent = False
    print(p, q, "\t", implies(p, q) and implies(q, p), "\t", int(p == q


if equivalent:
  print("\nexpressions are eqivalent")
else:
  print("\nexpressions not are eqivalent")

```

```
p q   ¬ (p → q)   p ∧ ¬q
0 0      1        1
0 1      0        0
1 0      0        0
1 1      1        1

expressions are eqivalent
```

d. (p → q) ∧ (p → r) and p→ (q ∧ r)

In [70]:

```python
def implies(p, q):
  if p == True and q == False:
    return 0
  return 1

equivalent = True
print("p q r (p → q)∧(p → r)  p→ (q ∧ r)")

for p in range(2):
  for q in range(2):
    for r in range(2):
      if implies(p, q) and implies(p, r) != implies(p, q and r):
        equivalent = False
      print(p, q, r ,"\t", implies(p, q) and implies(p, r), "\t ""\t",


if equivalent:
  print("\nexpressions are eqivalent")
else:
  print("\nexpressions not are eqivalent")

```

```
p q r (p → q)∧(p → r)  p→ (q ∧ r)
0 0 0    1               1
0 0 1    1               1
0 1 0    1               1
0 1 1    1               1
1 0 0    0               0
1 0 1    0               0
1 1 0    0               0
1 1 1    1               1

expressions are eqivalent
```

9 Tautology:

a. p∨¬p

In [59]:

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

def double_implies(p, q):
  return implies(p, q) and implies(q, p)

print("p\tp v ~p")

for p in [False, True]:
  print(p, '\t', p or (not p))
print()

```

```
p       p v ~p
False    True
True     True
```

b. ~ (a ∧ b) ↔ (~a ∨ ~b)

In [72]:

```
print('a\tb\t~(a ∧ b)↔(~a ∨ ~b)')
for a in [False, True]:
  for b in [False, True]:
    print(a, '\t', b, '\t', double_implies(not (a and b), (not a) or (n
print()

```

```
a       b       ~(a ∧ b)↔(~a ∨ ~b)
False    False   1
False    True    1
True     False   1
True     True    1
```

c. (((a ∨ b) ∧ (a → c)) ∧ (b → c)) → c

In [61]:

```
print('a\tb\tc\t(((a v b) ^ (a -> c)) ^ (b -> c)) -> c')
for a in [False, True]:
  for b in [False, True]:
    for c in [False, True]:
      print(a, '\t', b, '\t', c, '\t', implies(((a or b) and implies(a,
print()
```

```
a       b       c       (((a v b) ^ (a -> c)) ^ (b -> c)) -> c
False    False    False    True
False    False    True     True
False    True     False    True
False    True     True     True
True     False    False    True
True     False    True     True
True     True     False    True
True     True     True     True
```

In [ ]:

In [ ]: