**4.CRYPT-ARITHMETIC PROGRAM**from itertools import permutations

for p in permutations(range(10), 8):

s = dict(zip('SENDMORY', p))

if s['S'] == 0 or s['M'] == 0:

continue

send = 1000\*s['S'] + 100\*s['E'] + 10\*s['N'] + s['D']

more = 1000\*s['M'] + 100\*s['O'] + 10\*s['R'] + s['E']

money = 10000\*s['M'] + 1000\*s['O'] + 100\*s['N'] + 10\*s['E'] + s['Y']

if send + more == money:

print("SEND:", send, "MORE:", more, "MONEY:", money)

print("Mapping:", s)

Break

**6.VACCUM CLEANER**

loc, L, R, a = 'left', 'dirty', 'dirty', []

if loc == 'left':

if L=='dirty': a+=['Suck']; L='clean'

a+=['Move Right']; loc='right'

if R=='dirty': a+=['Suck']; R='clean'

print(a)

print({'left': L, 'right': R})

**12.TIC-TAC-TOE**board = [[' ']\*3 for \_ in range(3)]

print("Enter X or O for each cell (row 0-2, column 0-2):")

for i in range(3):

for j in range(3):

val = input(f"({i},{j}) X or O: ").strip().upper()

board[i][j] = val

print("\nTic Tac Toe Board:")

for row in board:

print(' | '.join(row))

print('-' \* 5) **15.Decision making tree**weather = input("Enter the weather (sunny/rainy): ")

temp = input("Enter the temperature (cool/hot): ")

if weather.lower() == "sunny":

decision = "Play" if temp.lower() == "cool" else "Don't Play"

elif weather.lower() == "rainy":

decision = "Play"

else:

decision = "Unknown"

print("Decision:", decision)

**16.FEED FORWARD NEURAL NETWORK**import numpy as np

X = [1, 0]

w1 = np.array([[0.5, 0.3], [0.4, 0.7]])

w2 = np.array([0.6, 0.9])

sig = lambda x: 1 / (1 + np.exp(-x))

h = sig(np.dot(X, w1))

out = sig(np.dot(h, w2))

print("Output:", out)

**17.SUM OF INTEGERS FROM 1 TO N:**

sum(1,1).

sum(N,Total):-

N>1,

N1 is N-1,

sum(N1,Temp),

Total is Temp+N.

sum(6,Y).

**18.DOB:**

name\_dob(john,15,april,1995).

name\_dob(alicxe,7,august,1975).

name\_dob(john,15,april,1995).

name\_dob(john,15,april,1995).

name\_dob(A,B,C,1975).

**19.STUDENT-TEACHER-SUB-CODE**student(john, cs101).

student(alice, cs102).

student(ravi, cs101).

student(sita, cs103).

teacher(dr\_smith, cs101).

teacher(ms\_anu, cs102).

teacher(mr\_khan, cs103).

subject(cs101, 'AI').

subject(cs102, 'DBMS').

subject(cs103, 'Networks').

student(john, Code), teacher(Teacher, Code).

**20.PLANETS DB**planet(mercury, 1, 4879, 0, terrestrial).

planet(venus, 2, 12104, 0, terrestrial).

planet(earth, 3, 12756, 1, terrestrial).

planet(mars, 4, 6792, 2, terrestrial).

planet(jupiter, 5, 142984, 79, gas\_giant).

planet(saturn, 6, 120536, 83, gas\_giant).

planet(uranus, 7, 51118, 27, ice\_giant).

planet(neptune, 8, 49528, 14, ice\_giant).

planet(earth, W, Y, C, N).  
  
**21.TOWERS OF HANOI**hanoi(1,A,B,\_):-

write('move disk from '),write(A),write(' to '),write(B),nl.

hanoi(N,A,B,C):-

N>1,

M is N-1,

hanoi(M,A,C,B),

hanoi(1,A,B,\_),

hanoi(M,C,B,A).

hanoi(3, r, c, l).

**22.BIRD FLY OR NOT**f(sparrow).

f(pigeon).

f(eagle).

s(X) :- f(X), write(X), write(' can fly'), nl.

s(X) :- \+f(X), write(X), write(' cannot fly'), nl.

s(sparrow) ,s(penguin).  
  
**23.PARENT RECOGNITION**male(john).

male(bob).

female(mary).

parent(john, bob).

parent(mary, bob).

father(X, Y) :- parent(X, Y).

mother(X, Y) :- parent(X, Y).

father(X,bob).  
  
**24.DIET MENU**disease\_diet(diabetes, 'Low sugar, high fiber, complex carbs').

disease\_diet(hypertension, 'Low salt, more fruits and vegetables').

disease\_diet(anemia, 'Iron-rich foods like spinach, red meat, beans').

disease\_diet(obesity, 'Low fat, high protein, portion control').

disease\_diet(gastritis, 'Soft foods, avoid spicy and acidic items').

disease\_diet(kidney\_stone, 'Drink more water, avoid oxalate-rich foods').

disease\_diet(diabetes, Diet).

**25.BANANA MONKEY PROBLEM**banana(state(\_, \_, \_, yes)) :-

write('Monkey got the banana!'), nl.

banana(state(\_, \_, no, no)) :-

write('Monkey moves, pushes box, climbs, and gets banana.'), nl,

banana(state(\_, \_, yes,no)).

banana(state(\_, \_, yes, no)) :-

write('Monkey climbs box and grabs banana.'), nl,

banana(state(\_, \_, \_, yes)).

banana(state(\_, \_, no, no)).

**26.FRUITS COLOURING**fruit\_color(apple, red).

fruit\_color(banana, yellow).

fruit\_color(grape, purple).

fruit\_color(orange, orange).

fruit\_color(kiwi, green).

fruit\_color(mango, yellow).

fruit\_color(blueberry, blue).

fruit\_color(apple, Color).

**27.BFS**edge(a,b).

edge(a,c).

edge(b,d).

edge(c,e).

edge(d,f).

bfs(G, [[G|P]|\_]) :- reverse([G|P], R), write(R), nl.

bfs(G, [[N|P]|R]) :- edge(N,X), \+ member(X,[N|P]),

append(R, [[X,N|P]], Q), bfs(G, Q).

bfs(S, G) :- bfs(G, [[S]]).

bfs(a, f).  
  
**28.PATIENT DIAGONSIS**disease(flu) :- s(fever), s(cough), s(body\_ache).

disease(cold) :- s(cough), s(sneezing), s(runny\_nose).

disease(malaria) :- s(fever), s(chills), s(sweating).

disease(typhoid) :- s(fever), s(abdominal\_pain), s(weakness).

s(S) :- write('Do you have '), write(S), write('? (yes/no): '), read(yes).

start :- disease(D), write('You may have: '), write(D), nl.

start.  
  
**29.FORWARD**fact(a).

fact(b).

fact(e).

rule(c) :- fact(a), fact(b).

rule(d) :- rule(c), fact(e).

forward :-

rule(X),

write('Derived from : '), write(X), nl,

fail.

forward.

forward.

**30.BACKWARD**known(a).

known(b).

goal(c) :- known(a), known(b).

goal(d) :- goal(c).

goal(d).

**31.NUMBER OF VOWELS**vowel(a). vowel(e). vowel(i). vowel(o). vowel(u).

count([], 0).

count([H|T], N) :-

vowel(H),

count(T, N1),

N is N1 + 1.

count([H|T], N) :-

\+ vowel(H),

count(T, N).

count([h,e,l,l,o], N).

**32,PATTERN MATCCHING**match(Pattern, List) :-

append(\_, Tail, List),

append(Pattern, \_, Tail).

match([b,c], [a,b,c,d]).