4.CRYPT-ARITHMETIC PROGRAM
```
from itertools import permutations
letters = 'SENDMORY'
for p in permutations(range(10), len(letters)):
    s = dict(zip(letters, p))
    if s['S'] == 0 or s['M'] == 0:
        continue
    send = s['S']*1000 + s['E']*100 + s['N']*10 + s['D']
    more = s['M']*1000 + s['O']*100 + s['R']*10 + s['E']
    money = s['M']*10000 + s['O']*1000 + s['N']*100 + s['E']*10 + s['Y']
    if send + more == money:
        print("SEND:", send, "MORE:", more, "MONEY:", money)
        print("Mapping:", s)
        break
```

12.TIC-TAC-TOE
```
board = [[' ' for _ in range(3)] for _ in range(3)]
print("Enter X or O for each position (row 0-2, column 0-2):")
for i in range(3):
    for j in range(3):
        while True:
            val = input(f"Enter X or O for position ({i}, {j}): ").upper()
            if val in ['X', 'O']:
                board[i][j] = val
                break
            else:
                print("Invalid input. Enter only X or O.")
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)
```

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

```
can_get_banana(state(_, _, _, yes)) :-
    write('Monkey got the banana!'), nl.

can_get_banana(state(_, _, no, no)) :-
    write('Monkey moves, pushes box, climbs, and gets banana.'), nl,
    can_get_banana(state(_, _, yes, yes)).

can_get_banana(state(_, _, yes, yes)) :-
    write('Monkey climbs box and grabs banana.'), nl.

can_get_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).

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

forward :-
   rule(X),
   write('Derived: '), 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_vowels([], 0).
count_vowels([H|T], N) :-
   vowel(H),
   count_vowels(T, N1),
   N is N1 + 1.
count_vowels([H|T], N) :-
   \+ vowel(H),
   count_vowels(T, N).

count_vowels([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]).
```