**TIC TAC TOE:**

import random

def pb(b): print("\n"+"\n---------\n".join(" | ".join(r) for r in b)+"\n")

cw=lambda b,p:any(all(c==p for c in r)for r in b)or any(all(b[i][j]==p for i in range(3))for j in range(3))or all(b[i][i]==p for i in range(3))or all(b[i][2-i]==p for i in range(3))

idraw=lambda b:all(c!=' 'for r in b for c in r)

def hm(b,p):

while 1:

try:

n=int(input(f"Player {p}, enter cell (1-9): "));r,c=(n-1)//3,(n-1)%3

if 1<=n<=9 and b[r][c]==' ':b[r][c]=p;break

print("Cell taken!")

except:print("Invalid!")

def cm(b,p):

print(f"Computer ({p}) is making a move...")

r,c=random.choice([(i,j)for i in range(3)for j in range(3)if b[i][j]==' ']);b[r][c]=p

def play():

b=[[' ']\*3 for \_ in range(3)];print("Welcome to Tic-Tac-Toe!")

m=input("Choose mode:\n1. Player vs Player\n2. Player vs Computer\nEnter 1 or 2: ");p1,p2,cur='X','O','X'

while 1:

pb(b);(hm if m=='1' or cur==p1 else cm)(b,cur)

if cw(b,cur):pb(b);print("Computer wins!" if m=='2'and cur==p2 else f"Player {cur} wins!");break

if idraw(b):pb(b);print("It's a draw!");break

cur=p2 if cur==p1 else p1

play()

**Water jug Problem**

from collections import deque

def water\_jug(j1,j2,t):

v,p,q={(0,0)}, {}, deque([(0,0)])

while q:

x,y=q.popleft()

if x==t or y==t:

path=[]

while (x,y)!=(0,0):path.append((x,y));x,y=p[(x,y)]

path.append((0,0));path.reverse()

print("Path to reach the target:")

for a,b in path:print(a,b)

return

for s in[(j1,y),(x,j2),(0,y),(x,0),

(x-min(x,j2-y),y+min(x,j2-y)),

(x+min(y,j1-x),y-min(y,j1-x))]:

if s not in v:v.add(s);p[s]=(x,y);q.append(s)

print("Solution not possible")

j1=int(input("Enter capacity of Jug 1: "))

j2=int(input("Enter capacity of Jug 2: "))

t=int(input("Enter target amount to measure: "))

water\_jug(j1,j2,t)

**N queens problem**

N=0;sol=[]

def prt(b,c):

print(f"Solution {c}:")

for r in b:print(" ".join("Q"if x else"."for x in r))

print()

def safe(b,r,c):

return not any(b[r][i]for i in range(c))and \

not any(b[i][j]for i,j in zip(range(r,-1,-1),range(c,-1,-1)))and \

not any(b[i][j]for i,j in zip(range(r,N),range(c,-1,-1)))

def solve(b,c):

if c>=N:sol.append([r[:]for r in b]);prt(b,len(sol));return

for i in range(N):

if safe(b,i,c):b[i][c]=1;solve(b,c+1);b[i][c]=0

def main():

global N;N=int(input("Enter board size (1 to 8): "))

if N<1 or N>8:print("Invalid input! Board size must be between 1 and 8.");return

if N==1:print("Solution 1:\n Q \n");print("Total Solutions for N = 1: 1");return

if N in(2,3):print(f"No solutions exist for N = {N}.");print(f"Total Solutions for N = {N}: 0");return

solve([[0]\*N for \_ in range(N)],0)

print(f"Total Solutions for N = {N} is {len(sol)}")

main()

**Missionaries and cannibels**

lM,lC,rM,rC,mv=3,3,0,0,0

prt=lambda d:print("\n"+"M "\*lM+"C "\*lC+f"| {d} |"+"M "\*rM+"C "\*rC+"\n")

def over():

if(lM and lC>lM)or(rM and rC>rM):print("Cannibals eat missionaries. You lost!");return 1

def ask(sM,sC):

while 1:

m,c=int(input("Missionaries: ")),int(input("Cannibals: "))

if 0<m+c<=2 and m<=sM and c<=sC:return m,c

print("Invalid move.")

print("\nMissionaries & Cannibals Game Start\nGoal: Move all to the right side safely.")

prt("---")

try:

while 1:

m,c=ask(lM,lC);lM-=m;lC-=c;rM+=m;rC+=c;mv+=1;prt("-->")

if rM+rC==6:print("You won in",mv,"moves! ");break

if over():break

m,c=ask(rM,rC);rM-=m;rC-=c;lM+=m;lC+=c;mv+=1;prt("<--")

if over():break

except:print("Input error.")

**Travelling Sales man Problem**

from itertools import permutations

n=int(input("Enter number of cities: "))

print("Enter the distance matrix (row by row):")

d=[list(map(int,input().split()))for \_ in range(n)]

start=0;path=None;cost=float("inf")

for p in permutations([i for i in range(n) if i!=start]):

c=sum(d[a][b] for a,b in zip((start,)+p,p+(start,)))

if c<cost:cost,path=c,(start,)+p+(start,)

print("Shortest Path:",path)

print("Minimum Cost:",cost)

**Both AO\* and A\***

def ao(n,g,h,s,p):

print(f"\nExpanding: {n}")

if s.get(n): return h[n]

if n not in g: s[n]=1; return h[n]

c={}

for r,ch in g[n]:

if r=="OR": c[f"OR({ch[0]})"]=h[ch[0]]+1

else: c[f"AND({','.join(ch)})"]=sum(h[x]+1 for x in ch)

b=min(c,key=c.get); h[n]=c[b]

print("Choices:",c,"Best:",b,"Revised h =",h[n])

for x in ([b[4:-1].split(",")] if "AND" in b else [[b[3:-1]]])[0]:

p[x]=n; ao(x,g,h,s,p)

s[n]=1; return h[n]

def run\_ao():

nodes=input("Node names: ").split()

h={x:int(input(f"h({x})=")) for x in nodes}; g={}

print("Edges Parent AND/OR child1,child2 (done to stop):")

while 1:

l=input()

if l=="done": break

u,r,ch=l.split(); g.setdefault(u,[]).append((r.upper(),ch.split(",")))

st=input("Start node: "); print("\n--- AO\* Started ---")

ao(st,g,h,{},{}); [print(f"h({x})={h[x]}") for x in nodes]; print("Start node cost =",h[st])

def a\_star(g,st,gl,h):

G={x:1e9 for x in g};P={x:None for x in g}; G[st]=0;O=[st]

while O:

n=min(O,key=lambda x:G[x]+h[x]); O.remove(n)

print("Visiting:",n,"g=",G[n],"h=",h[n],"f=",G[n]+h[n])

if n==gl:

print("Destination reached!"); path=[]

while n: path.append(n); n=P[n]

print("Path:"," -> ".join(path[::-1])); [print(x,":",G[x]+h[x]) for x in g]; return

for v,w in g[n]:

new\_g=G[n]+w

if new\_g<G[v]: G[v]=new\_g; P[v]=n;

if v not in O: O.append(v)

def run\_a():

nodes=input("Node names: ").split(); nodes=[x.strip() for x in nodes] # strip spaces

g={x:[] for x in nodes}

print("Edges u v w (done to stop):")

while 1:

l=input()

if l=="done": break

u,v,w=l.split()

u,v=u.strip(),v.strip() # strip spaces

g[u].append((v,int(w))); g[v].append((u,int(w)))

h={x:int(input(f"h({x})=")) for x in nodes}

st=input("Start:").strip() # strip spaces

gl=input("Goal:").strip() # strip spaces

a\_star(g,st,gl,h)

while 1:

c=input("\n1.AO\*\n2.A\*\n3.Exit\nChoice: ")

if c=="1": run\_ao()

elif c=="2": run\_a()

elif c=="3": break

**AO\***

def ao(n,g,h,s,p):

print(f"\nExpanding: {n}")

if s.get(n): return h[n]

if n not in g: s[n]=1; return h[n]

c={};

for r,ch in g[n]: c[f"{r}({','.join(ch)})"]=sum(h[x]+1 for x in ch) if r=="AND" else h[ch[0]]+1

b=min(c,key=c.get); h[n]=c[b]; print("Choices:",c,"Best:",b,"Revised h =",h[n])

for x in ([b[4:-1].split(",")] if "AND" in b else [[b[3:-1]]])[0]: p[x]=n; ao(x,g,h,s,p)

s[n]=1; return h[n]

nodes=input("Node names:").split(); nodes=[x.strip() for x in nodes]

h={x:int(input(f"h({x})=")) for x in nodes}; g={}

print("Edges Parent AND/OR child1,child2 (done to stop):")

while 1:

l=input();

if l=="done": break

u,r,ch=l.split(); g.setdefault(u.strip(),[]).append((r.upper(),[x.strip() for x in ch.split(",")]))

st=input("Start node:").strip(); print("\n--- AO\* Started ---")

ao(st,g,h,{},{}); [print(f"h({x})={h[x]}") for x in nodes]; print("Start node cost =",h[st])

**A\***

def a\_star(g,st,gl,h):

G={x:1e9 for x in g}; P={x:None for x in g}; G[st]=0; O=[st]

while O:

n=min(O,key=lambda x:G[x]+h[x]); O.remove(n)

print("Visiting:",n,"g=",G[n],"h=",h[n],"f=",G[n]+h[n])

if n==gl:

print("Destination reached!"); path=[]

while n:path.append(n);n=P[n]

print("Path:"," -> ".join(path[::-1])); [print(x,":",G[x]+h[x]) for x in g]; return

for v,w in g[n]:

ng=G[n]+w

if ng<G[v]: G[v]=ng; P[v]=n; O+=[v] if v not in O else []

nodes=input("Node names:").split(); nodes=[x.strip() for x in nodes]; g={x:[] for x in nodes}

print("Edges u v w (done to stop):")

while 1:

l=input()

if l=="done": break

u,v,w=l.split(); u,v=u.strip(),v.strip(); g[u].append((v,int(w))); g[v].append((u,int(w)))

h={x:int(input(f"h({x})=")) for x in nodes}

st=input("Start:").strip(); gl=input("Goal:").strip(); a\_star(g,st,gl,h)