**PROGRAM**

**2 Jugs Problem**

jugA=int(input("Enter the capacity of jug A:::"))

jugB=int(input("Enter the capacity of jug B:::"))

jugAi=int(input("Enter the initial state of jug A:::"))

jugBi=int(input("Entere the initial state of jug B:::"))

jugAf=int(input("Enter the Final state of jug A:::"))

print("Operation 1:Fill the Jug A completely")

print("Operation 2:Fill the Jug B completely")

print("Operation 3:Empty the Jug A compltely on ground")

print("Operation 4:Empty the Jug B compltely on ground ")

print("Operation 5:Pour the water from Jug A to Jug B completely or until Jug A becomes empty")

print("Operation 6:Pour the water from Jug B to Jug A completely or until Jug B becomes empty ")

print("Operation 7:Pour all the water from Jug B to Jug A ")

print("Operation 8:Pour all the water from Jug A to Jug B ")

while(jugAi!=jugAf):

op=int(input("Enter the Operation Number:::"))

if op==1:

jugAi=jugA

if op==2:

jugBi=jugB

if op==3 :

jugAi=0;

if op==4:

jugBi=0

if op==5:

if jugB-jugBi >jugAi :

jugBi=jugBi+jugAi

jugAi=0

else:

jugAi=jugAi-(jugB-jugBi)

jugBi=jugB

if op==6:

if jugA-jugAi >jugBi :

jugAi=jugAi+jugBi

jugBi=0

else:

jugBi=jugBi-(jugA-jugAi)

jugAi=jugA

if op==7:

jugAi=jugAi+jugBi

jugBi=0

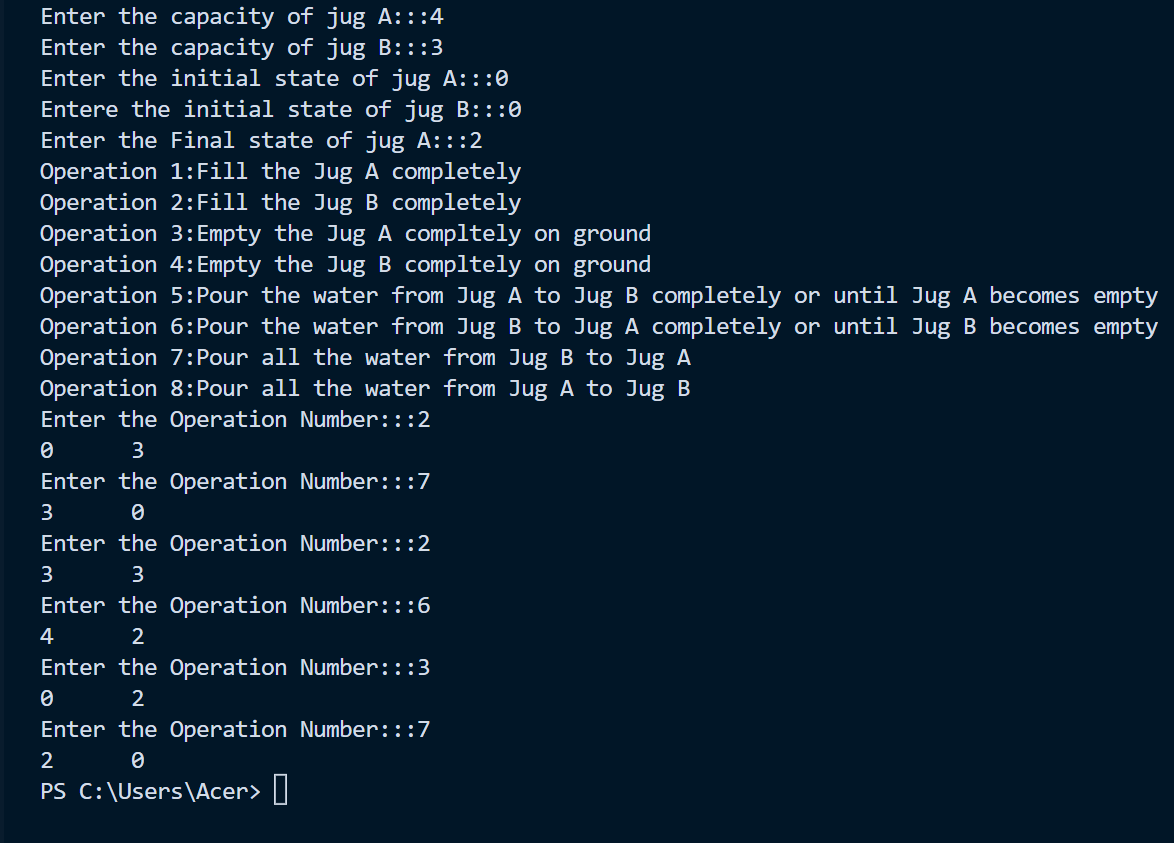
if op==8:

jugBi=jugBi+jugAi

jugAi=0

print(jugAi," ",jugBi)

**OUTPUT**

****

**PROGRAM**

**3 Jugs Problem**

x=int(input("Enter the capacity of jug A:::"))

y=int(input("Enter the capacity of jug B:::"))

z=int(input("Enter the capacity of jug C:::"))

goalA=int(input("Enter the goal state of jug A:::"))

goalB=int(input("Enter the goal state of jug B:::"))

initial\_state=(8,0,0)

visited={} #all visited

path=[] #To store entire path

def waterJugState(state):

a=state[0]

b=state[1]

c=state[2]

if(a==goalA and b==goalB):

path.append(state)

return True

if((a,b,c) in visited):

return False

visited[(a,b,c)]=1

if(a>0):

if(a+b<=y):

if(waterJugState((0,a+b,c))):

path.append(state)

return True

else:

if(waterJugState((a-(y-b),y,c))):

path.append(state)

return True

if(a+c<=z):

if(waterJugState((0,b,a+c))):

path.append(state)

return True

else:

if(waterJugState((a-(z-c),b,z))):

path.append(state)

return True

if(b>0):

if(b+a<=x):

if(waterJugState((a+b,0,c))):

path.append(state)

return True

else:

if(waterJugState((x,b-(x-a),c))):

path.append(state)

return True

if(b+c<=z):

if(waterJugState((a,0,b+c))):

path.append(state)

return True

else:

if(waterJugState((a,b-(z-c),z))):

path.append(state)

return True

if(c>0):

if(c+a<=x):

if(waterJugState((a+c,b,0))):

path.append(state)

return True

else:

if((waterJugState(x,b,c-(x-a)))):

path.append(state)

return True

if(c+b<=y):

if(waterJugState((a,b+c,0))):

path.append(state)

return True

else:

if(waterJugState((a,y,c-(y-b)))):

path.append(state)

return True

return False

waterJugState(initial\_state)

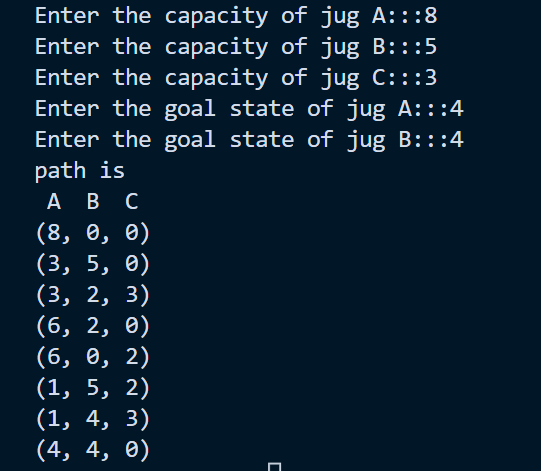
print("path is")

print(" A B C")

path.reverse()

for i in path:

print(i)

**OUTPUT**