#Water Jug Problem

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

b = int(input("Enter the capacity of Jug B: "))

ai = int(input("enter the initial water in Jug A: ")) #going to change everytime

bi = int(input("Enter the initial water in Jug B: "))

bf = int(input("Enter the final capacity of Jug B: "))

af = int(input("Enter the final capacity of Jug A: "))

print("List of operations you can perform are: ")

print("Op1: Fill the Jug A completely")

print("Op2: Fill the Jug B completely")

print("Op3: Empty the Jug A completely")

print("Op4: Empty the Jug B completely")

print("Op5: Pour water forom Jug A to B until it is full ")

print("Op6: Pour water forom Jug B to A until it is full ")

print("Op7: Pour all the water from Jug A to B ")

print("Op8: Pour all the water from Jug B to A ")

while( ai != af or bi != bf):

op = int(input("Enter the operation number: "))

if op == 1:

ai = a

elif op == 2:

bi = b

elif op == 3:

ai = 0

elif op == 4:

bi = 0

elif op == 5:

if (b - bi > ai):

bi = bi + ai

ai = 0

else:

ai = ai - (b-bi)

bi = b

elif op == 6:

if (a - ai > bi):

ai = ai + bi

bi = 0

else:

bi = bi - (a - ai)

ai = a

elif op == 7:

bi = ai + bi

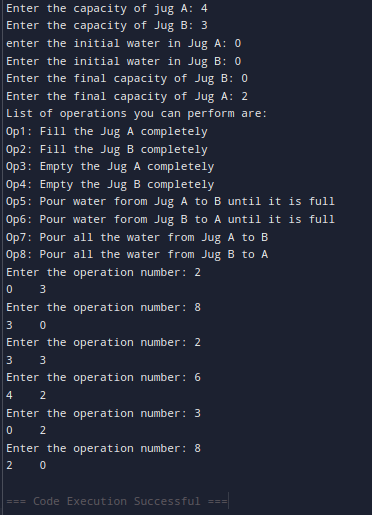
ai = 0

elif op == 8:

ai = ai + bi

bi = 0

print(ai, "\t", bi)

OUTPUT:  


#Water Jug Problem (3 Jugs)

x = int(input('Enter the maximum capacity of Jug A:'))

y = int(input('Enter the maximum capacity of Jug B:'))

z = int(input('Enter the maximum capacity of Jug C:'))

goalx = int(input('Enter the goal state in Jug A:'))

goaly = int(input('Enter the goal state in Jug B:'))

visited = {} #To mark visited states

path = [] #Store solution path

def all\_states(state):

jugA, jugB, jugC = state

if (jugA == goalx and jugB == goaly):

path.append(state)

return True

if(state in visited):

return False

visited[state] = 1

if (jugA > 0):

if (jugA + jugB <= y):

if (all\_states((0, jugA + jugB, jugC))):

path.append(state)

return True

else:

if (all\_states((jugA - (y - jugB), y, jugC))):

path.append(state)

return True

if(jugA + jugC <= z):

if (all\_states((0, jugB, jugA + jugC))):

path.append(state)

return True

else:

if(all\_states((jugA - (z - jugC), jugB, z))):

path.append(state)

return True

if(jugB > 0):

if(jugA + jugB <= x):

if(all\_states((jugA + jugB, 0, jugC))):

path.append(state)

return True

else:

if (all\_states((x, jugB - (x - jugA), jugC))):

path.append(state)

return True

if(jugB + jugC <= z):

if(all\_states((jugA, 0, jugB + jugC))):

path.append(state)

return True

else:

if(all\_states((jugA, jugB - (z - jugC), z))):

path.append(state)

return True

if(jugC > 0):

if(jugA + jugC <= x):

if(all\_states((jugA + jugC, jugB, 0))):

path.append(state)

return True

else:

if(all\_states((x, jugB, jugC - (x - jugA)))):

path.append(state)

return True

if(jugB + jugC <= y):

if(all\_states((jugA, jugB + jugC, 0))):

path.append(state)

return True

else:

if (all\_states((jugA, y, jugC - (y - jugB)))):

path.append(state)

return True

return False

initial\_state = (x, 0, 0)

print("Starting work...\n")

all\_states(initial\_state)

path.reverse()

for state in path:

print(state)

OUTPUT:

