Name: Swapnil

Group: 2CS12

Roll Number: 102016108

Artificial Intelligence (AI) UCS411

Assignment-2

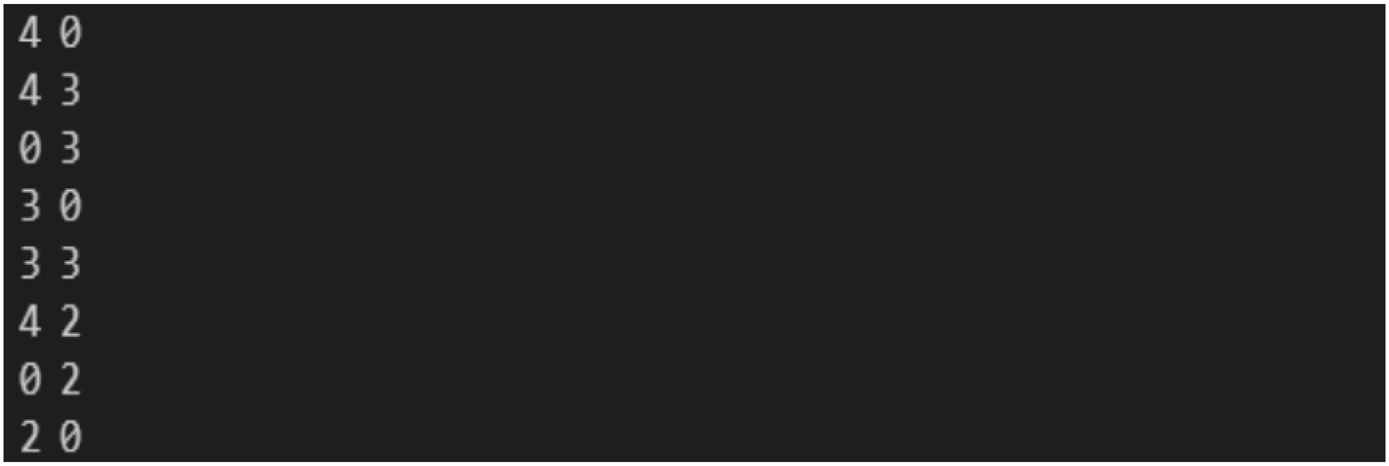
Question-1: Given two jugs- a 4 litre and 3 litre capacity. Neither has any measurable markers on it. There is a pump which can be used to fill the jugs with water. Simulate the procedure in Python to get exactly 2 litre of water into 4-litre jug.

Code:

```
x = 0
```

```
y = 0

state = [[0, 0]]

while(x != 2):

    if [4, y] not in state and x < 4:

        x = 4

        state.append([x, y])

        print(x, y)

    if [x, 3] not in state and y < 3:

        y = 3

        state.append([x, y])

        print(x, y)

    if [0, y] not in state and x > 0:

        x = 0

        state.append([x, y])

        print(x, y)

    if [x, 0] not in state and y > 0:

        y = 0

        state.append([x, y])

        print(x, y)

    if [4, y-(4-x)] not in state and x+y >= 4 and y > 0:

        y = y-(4-x)

        x = 4

        state.append([x, y])

        print(x, y)

    if [x-(3-y), 3] not in state and x+y >= 3 and x > 0:

        x = x-(3-y)

        y = 3

        state.append([x, y])

        print(x, y)
    if [x+y, 0] not in state and (x+y) <= 4 and y >= 0:

        x = x+y

        y = 0

        state.append([x, y])

        print(x, y)

    if [0, y+x] not in state and (x+y) <= 3 and x >= 0:

        y = x+y

        x = 0

        state.append([x, y])

        print(x, y)
```

Output:



```
4 0
4 3
0 3
3 0
3 3
4 2
0 2
2 0
```

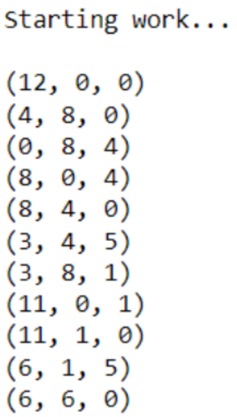
Question-2: Given three jugs: 12, 8 and 5 liter capacities. Largest jug is completely filled. Using these 3 jugs, split the water to obtain exactly 6 liter in largest jugs.

Code:

```
# 3 water jugs capacity -> (x,y,z) where x>y>z
# initial state (12,0,0)
# final state (6,6,0)
capacity = (12,8,5)
# Maximum capacities of 3 jugs -> x,y,z
x = capacity[0]
y = capacity[1]
z = capacity[2]
# to mark visited states
memory = {}
# store solution path
ans = []
def get_all_states(state):
    # Let the 3 jugs be called a,b,c
    a = state[0]
    b = state[1]
    c = state[2]
    if(a==6 and b==6):
        ans.append(state)
        return True
    # if current state is already visited earlier
    if((a,b,c) in memory):
        return False
```

```
memory[(a,b,c)] = 1

#empty jug a

if(a>0):

    #empty a into b

    if(a+b<=y):

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

            ans.append(state)

            return True

    else:

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

            ans.append(state)

            return True

    #empty a into c

    if(a+c<=z):

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

            ans.append(state)

            return True

else:

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

        ans.append(state)

        return True

#empty jug b

if(b>0):

#empty b into a

    if(a+b<=x):

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

            ans.append(state)

            return True

    else:
        if( get_all_states((x, b-(x-a), c)) ):
            ans.append(state)
            return True
#empty b into c
if(b+c<=z):
    if( get_all_states((a, 0, b+c)) ):
        ans.append(state)
        return True
else:
    if( get_all_states((a, b-(z-c), z)) ):
        ans.append(state)
        return True
#empty jug c
if(c>0):
    #empty c into a
    if(a+c<=x):
        if( get_all_states((a+c, b, 0)) ):
            ans.append(state)
            return True
    else:
        if( get_all_states((x, b, c-(x-a))) ):
            ans.append(state)
            return True
#empty c into b
    if(b+c<=y):
        if( get_all_states((a, b+c, 0)) ):
            ans.append(state)
            return True
else:
        if( get_all_states((a, y, c-(y-b))) ):
            ans.append(state)
            return True
    return False
initial_state = (12,0,0)
print("Starting work...\n")
get_all_states(initial_state)
ans.reverse()
for i in ans:
    print(i)
```

Output:



```
Starting work...

(12, 0, 0)
(4, 8, 0)
(0, 8, 4)
(8, 0, 4)
(8, 4, 0)
(3, 4, 5)
(3, 8, 1)
(11, 0, 1)
(11, 1, 0)
(6, 1, 5)
(6, 6, 0)
```

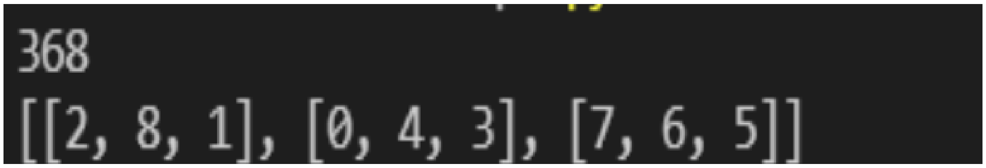
Question-3: Write a code in python for the 8 puzzle problem by taking the following initial and final states.

Code:

```python
import copy

start_state = [[1, 2, 3], [8, 0, 4], [7, 6, 5]]

goal_state = [[2, 8, 1], [0, 4, 3], [7, 6, 5]]

q = []

cnt = 1

def find_pos(start):

    for i in range(len(start)):

        for j in range(len(start[i])):

            if start[i][j] == 0:

                return (i, j)

def compare(curr, goal):

    if curr == goal:

        return 1

    else:

        return 0

def up(state, u, v):

    state[u][v] = state[u-1][v] # up

    state[u-1][v] = 0

    return state

def right(state, l, m):

    state[l][m] = state[l][m+1] # right

    state[l][m+1] = 0

    return state

def left(state, n, o):

    state[n][o] = state[n][o-1] # left

    state[n][o-1] = 0

    return state

def down(state, r, s):
    state[r][s] = state[r+1][s] # down
    state[r+1][s] = 0
    return state
def states(start, goal):
    pos = find_pos(start)
    i = pos[0]
    j = pos[1]
    global cnt
    if i-1 >= 0:
        state_1 = copy.deepcopy(start)
        state1 = up(state_1, i, j)
        if compare(state1, goal):
            print(cnt)
            print(state1)
            return 1
        else:
            if state1 not in q:
                q.append(state1)
                cnt += 1
    if j+1 <= 2:
        state_2 = copy.deepcopy(start)
        state2 = right(state_2, i, j)
        if compare(state2, goal):
            print(cnt)
            print(state2)
            return 1
        else:
            if state2 not in q:
                q.append(state2)
                cnt += 1
    if j-1 >= 0:
        state_3 = copy.deepcopy(start)
        state3 = left(state_3, i, j)
        if compare(state3, goal):
                print(cnt)
                print(state3)
                return 1
        else:
                if state3 not in q:
                    q.append(state3)
                    cnt += 1
        if i+1 <= 2:
            state_4 = copy.deepcopy(start)
            state4 = down(state_4, i, j)
            if compare(state4, goal):
                print(cnt)
                print(state4)
                return 1
            else:
                if state4 not in q:
                    q.append(state4)
                    cnt += 1
    def func():
        if not compare(start_state, goal_state):
            q.append(start_state)
    i = 0
    while(q):
        spop = q[i]
```

```
        i += 1
        if(states(spop, goal_state)):
            break
    func()
```

Output:



```
368
[[2, 8, 1], [0, 4, 3], [7, 6, 5]]
```

Question-4: Write a Python program to implement Travelling Salesman Problem (TSP). Take the starting node from the user at run time.

Code:

```
from sys import maxsize
from itertools import permutations
V = 4
def travellingSalesmanProblem(graph, s):
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
    min_path = maxsize
    all_perm = list(permutations(vertex))
    all_perm.append(vertex)
    for j in range(len(all_perm)):
        current_pathweight = 0
        k = s
        vertex = all_perm[j]
        for i in range(len(vertex)):
            current_pathweight += graph[k][vertex[i]]
            k = vertex[i]
        current_pathweight += graph[k][s]
        min_path = min(min_path, current_pathweight)
    return min_path
graph = [[0, 10, 15, 20], [10, 0, 35, 25],
[15, 35, 0, 30], [20, 25, 30, 0]]
# s = int(input())
s = 1
print(travellingSalesmanProblem(graph, s))
```

Output: