Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

1. The John is an engineering student of CS/IT and right now he is in the home during lockdown period in COVID-19 situation. The John is playing mobile games in all the time, therefore his mother requests him to perform a task with justification (proof of procedure followed to solve the following problem) for her. According to that task, she told:

   "I am giving you two buckets, a 4-litre one and a 3-litre one. Neither have any measuring markers on it. There is a pump that can be used to fill the buckets with water. How can you get exactly 2-litre water into the 4-litre bucket?"

   The John is an engineering student, and he has knowledge of searching algorithms. So, he is ready to solve the above task virtually (for justification purpose) by using BFS algorithm.

   Write a Python program to implement production rule systems of the above task for preparing the justification for his mother.

```
import collections

def main():

    starting_node = [[0, 0]]
    buckets = get_buckets()
    goal_amount = get_goal(buckets)
    check_dict = {}
    search(starting_node, buckets, goal_amount, check_dic
t)

def get_index(node):
    return pow(7, node[0]) * pow(5, node[1])


def get_buckets():
    print("Enter the volume of the Buckets...")
    buckets = []
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
    temp = int(input("\nEnter first bucket volume  :   ")
)
    while temp < 1:
        temp = int(input("\nEnter a valid amount   :   ")
)
    buckets.append(temp)

    temp = int(input("\nEnter second bucket volume :   ")
)
    while temp < 1:
        temp = int(input("\nEnter a valid amount   :   ")
)
    buckets.append(temp)

    return buckets

def get_goal(buckets):
    max_amount = max(buckets[0], buckets[1])
    s = "\nEnter the desired amount of water (1 - {0}) fo
r first bucket :  ".format(max_amount)
    goal_amount = int(input(s))
    while goal_amount < 1 or goal_amount > max_amount:
        goal_amount = int(input("\nEnter a valid amount (
1 - {0}): ".format(max_amount)))

    return goal_amount

def is_goal(path, goal_amount):

    return path[-1][0] == goal_amount

def been_there(node, check_dict):
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
    return check_dict.get(get_index(node), False)

def next_transitions(jugs, path, check_dict):


    result = []
    next_nodes = []
    node = []

    a_max = jugs[0]
    b_max = jugs[1]

    a = path[-1][0]
    b = path[-1][1]

    # 1. fill in the first bucket
    node.append(a_max)
    node.append(b)
    if not been_there(node, check_dict):
        next_nodes.append(node)
    node = []

    # 2. fill in the second bucket
    node.append(a)
    node.append(b_max)
    if not been_there(node, check_dict):
        next_nodes.append(node)
    node = []

    # 3. second bucket to first bucket
    node.append(min(a_max, a + b))
    node.append(b - (node[0] - a))  # b - ( a' - a)
    if not been_there(node, check_dict):
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
            next_nodes.append(node)
        node = []

        # 4. first bucket to second bucket
        node.append(min(a + b, b_max))
        node.insert(0, a - (node[0] - b))
        if not been_there(node, check_dict):
            next_nodes.append(node)
        node = []

        # 5. empty first bucket
        node.append(0)
        node.append(b)
        if not been_there(node, check_dict):
            next_nodes.append(node)
        node = []

        # 6. empty second bucket
        node.append(a)
        node.append(0)
        if not been_there(node, check_dict):
            next_nodes.append(node)

        for i in range(0, len(next_nodes)):
            temp = list(path)
            temp.append(next_nodes[i])
            result.append(temp)


        return result


def transition(old, new, buckets):
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
    a = old[0]
    b = old[1]
    a_prime = new[0]
    b_prime = new[1]
    a_max = buckets[0]
    b_max = buckets[1]

    if a > a_prime:
        if b == b_prime:
            return "Clear {0}-
litre bucket:\t\t\t".format(a_max)
        else:
            return "Pour {0}-litre bucket into {1}-
litre bucket:\t".format(a_max, b_max)
    else:
        if b > b_prime:
            if a == a_prime:
                return "Clear {0}-
litre bucket:\t\t\t".format(b_max)
            else:
                return "Pour {0}-litre bucket into {1}-
litre bucket:\t".format(b_max, a_max)
        else:
            if a == a_prime:
                return "Fill {0}-
litre bucket:\t\t\t".format(b_max)
            else:
                return "Fill {0}-
litre bucket:\t\t\t".format(a_max)


def print_path(path, buckets):
    print("Starting from:\t\t\t\t\t", path[0])
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
    for i in  range(0, len(path) - 1):
        print(i+1,":", transition(path[i], path[i+1], buc
kets), path[i+1])
    print("\nRequired goal reached")

def search(starting_node, buckets, goal_amount, check_dic
t):
    print("\n\n\t\t------Implementing BFS------\n\n")

    goal = []
    accomplished = False

    q = collections.deque()
    q.appendleft(starting_node)

    while len(q) != 0:
        path = q.popleft()
        check_dict[get_index(path[-1])] = True
        if len(path) >= 2:
            transition(path[-2], path[-
1], buckets), path[-1]
        if is_goal(path, goal_amount):
            accomplished = True
            goal = path
            break

        next_moves = next_transitions(buckets, path, chec
k_dict)
        for i in next_moves:
            q.append(i)

    if accomplished:
        print("Printing the sequence of the moves :- \n")
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
        print_path(goal, buckets)
    else:
        print("Problem cannot be solved.")


if __name__ == '__main__':
    main()
```

OUTPUT :

```
Enter the volume of the Buckets...

Enter first bucket volume  :   5

Enter second bucket volume :   6

Enter the desired amount of water (1 - 6) for first bucke
t :  4

    ------Implementing BFS------
Printing the sequence of the moves :-

Starting from:           [0, 0]
1 : Fill 5-litre bucket:       [5, 0]
2 : Pour 5-litre bucket into 6-litre bucket:   [0, 5]
3 : Fill 5-litre bucket:       [5, 5]
4 : Pour 5-litre bucket into 6-litre bucket:   [4, 6]

Required goal reached
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

2. Now, the John is working for a Water supply company. He has to present the solution for a given problem in front of his boss. He needs to be make presentation for that problem which is as follows:

“The task is to set up a connection for water supply. Set the water supply in one city and water gets transported from it to other cities using road transport. Certain cities are blocked which means that water cannot pass through that particular city. Determine the maximum number of cities to which water can be supplied.”

During presentation, he is using one of the uninformed search algorithms to solve the above problem with some raw data as: given N cities which are connected using N-1 roads. Between Cities [i, i+1], there exists an edge for all i from 1 to N-1. The following information have been used in the presentation:

· The first line contains an integer >strong>N denoting the number of cities.

· The next N-1 lines contain two space-separated integers u v denoting a road between city u and v.

· The next line contains N space-separated integers where it is 1 if the ith city is blocked, else it is 0.

Write a Python program to implement production rule systems for water supply problem.

```
def BFS_Water(v, visited, adj, src):

    visited[src] = True
    q = []
    q.append(src)
    count = 0

    while (len(q) != 0):
```

Name: Lakhan Kumawat

Roll No:1906055

Branch: CSE

Course Code: CSL5402

```
        p = q[0]

        for i in range(len(adj[p])):

            if (visited[adj[p][i]] == False and v[adj[p][
i]-1] == 0):
                count += 1
                visited[adj[p][i]] = True
                q.append(adj[p][i])

            elif(visited[adj[p][i]] == False and v[adj[p]
[i]-1] == 1):
                count += 1
        q.remove(q[0])

    return count + 1

def bfs(N, v, adj):
    visited = [ 0 for i in range(N + 1)]
    temp = 1

    for i in range(1, N + 1, 1):
        visited[i] = False

    for i in range(1, N + 1, 1):
        if (v[i-1] == 0 and visited[i-1] == False):
            res = BFS_Water(v, visited, adj, i)
            if (res > temp):
                temp = res

    return temp

N = int(input("\nEnter the no.of cities:- "))
```

Name: Lakhan Kumawat
Roll No:1906055
Branch: CSE
Course Code: CSL5402

```
adj = [[] for i in range(N + 1)]
v = [0 for i in range(N+1)]
print("\nenter values for denoting roads:- ")
for i in range(1,N):
    a,b=map(int,input().split())
    adj[a].append(b)
v=list(map(int,(input("\nEnter the cities are blocked or
not:- ").split())))

print("\nMaximun no.of cities supplied by water are:- ",b
fs(N, v, adj))
```

OUTPUT :

```
Enter the no.of cities:-  4

enter values for denoting roads:-
 1 2
 2 3
 3 4

Enter the cities are blocked or not:-  0 1 1 1

Maximun no.of cities supplied by water are:-  2
```