A.I LAB

ASSIGNMENT 1

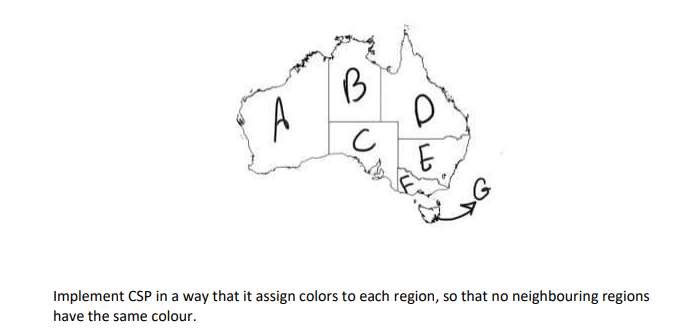
Name: Syed Muzzamil Waseem

SID: 11067

CID: 110089

Date: 04 August 2022

QUESTION 1:



**CODE:**

colors = ['Red','Blue', 'Green']

states = ['A', 'B', 'C', 'D', 'E', 'F', 'G']

neighbours={}

neighbours['A'] = ['B', 'C']

neighbours['B'] = ['A', 'C', 'D']

neighbours['C'] = ['A', 'B', 'D', 'E', 'F']

neighbours['D'] = ['B', 'C', 'E']

neighbours['E'] = ['C', 'D', 'F']

neighbours['F'] = ['C', 'E']

neighbours['G'] = ['']

colorofstate={}

def promising(state,color):

for neighbour in neighbours.get(state):

colorofneighbour = colorofstate.get(neighbour)

if colorofneighbour == color:

return False

return True

def colorforstate(state):

for color in colors:

if promising(state,color):

return color

return None

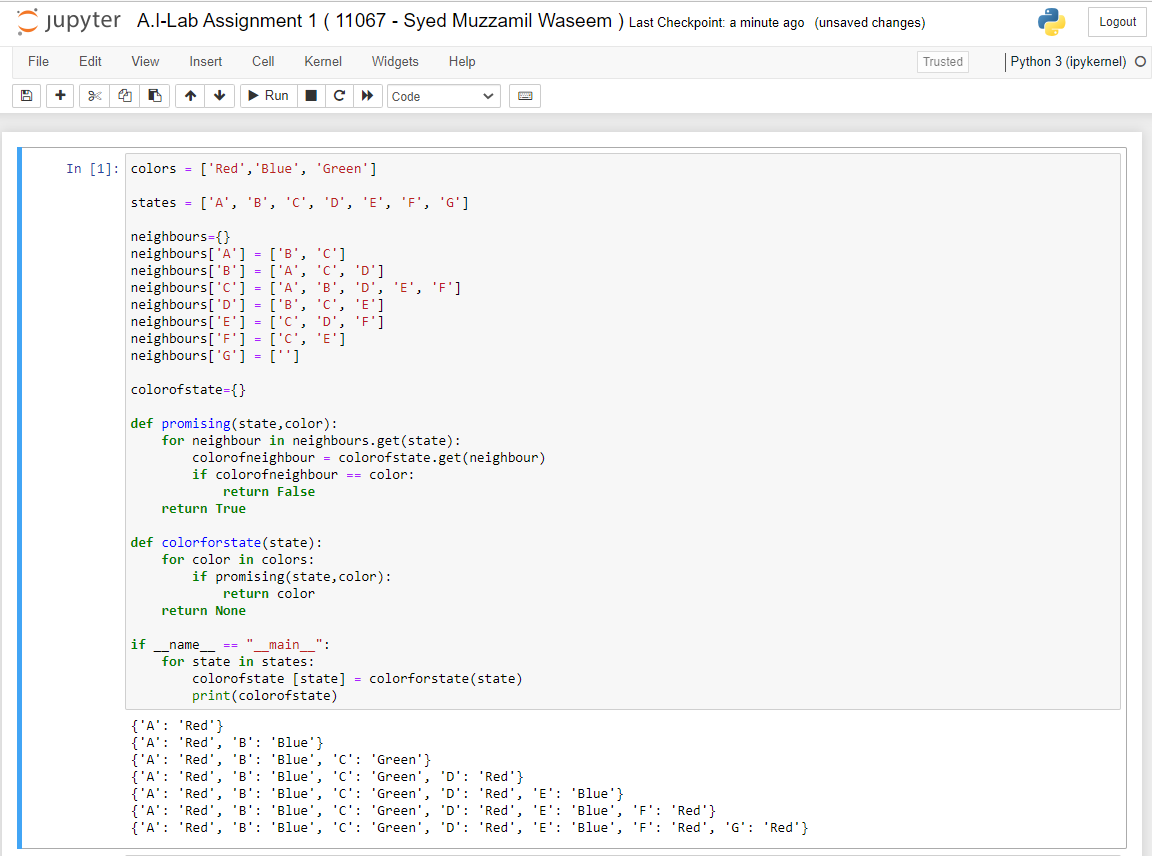
if \_\_name\_\_ == "\_\_main\_\_":

for state in states:

colorofstate [state] = colorforstate(state)

print(colorofstate)

OUTPUT:



QUESTION 2:

2. Implement simple reflex agent in which, the environment of Lab 1 and Lab 2 will be clean or dirty and the agent will done the action according to the given instruction.

**CODE:**

from random import randint

env=["Lab-1","Lab-2"]

acctuator=["This lab is clean.","This lab is dirty."]

def L1(i):

index=i

for a in env:

if index==0:

print("Lab-1:")

print(acctuator[index])

print("The vacuum cleaner is now moving in Lab-2...\n")

return L2(randint(0, 1))

elif index==1:

print("Lab-1:")

print(acctuator[index])

print("Lab-1 is getting cleaned...\n")

index=index-1

def L2(i):

index=i

for b in env:

if index==0:

print("Lab-2:")

return(acctuator[index])

elif index==1:

print("Lab-2:")

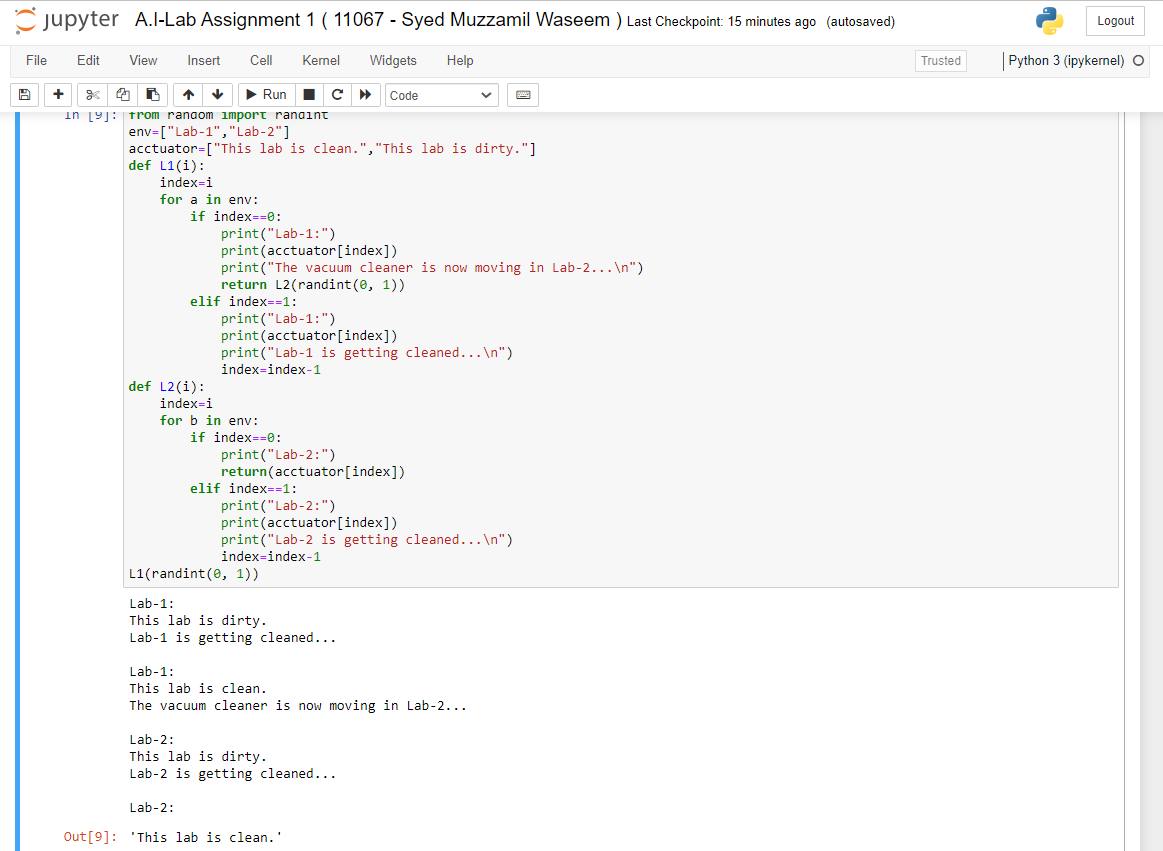
print(acctuator[index])

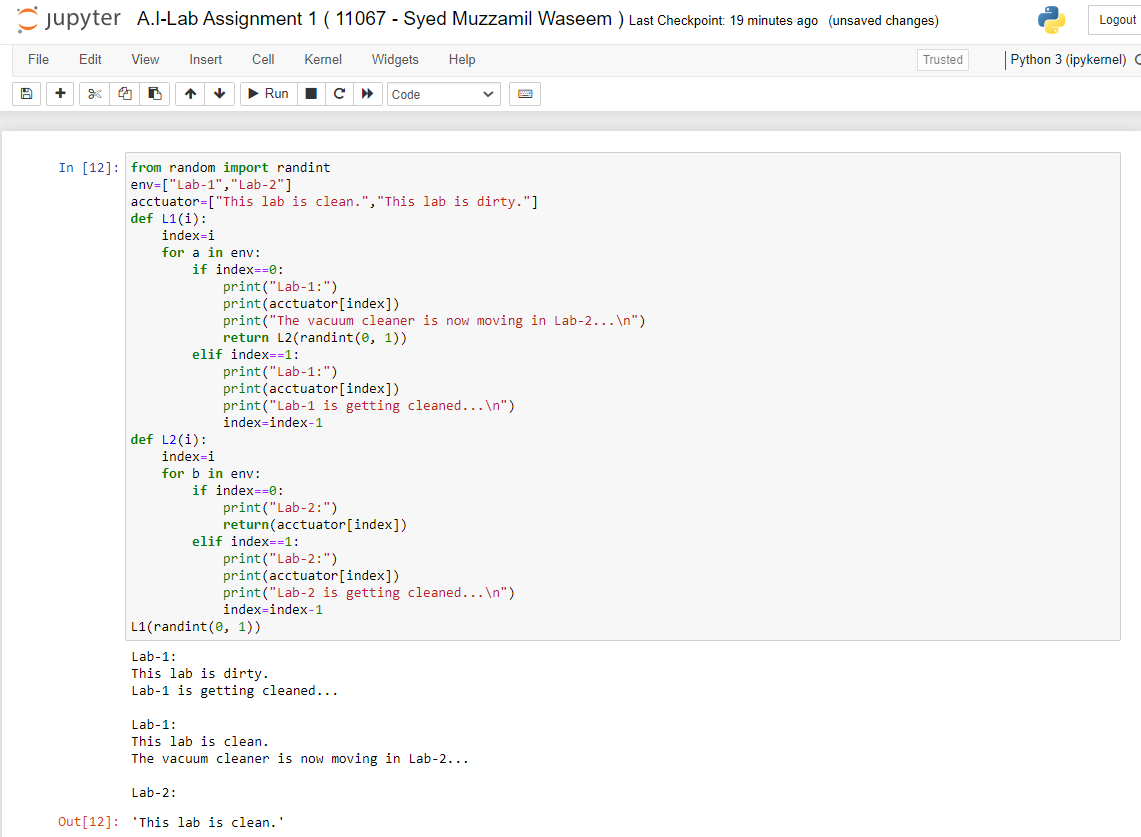
print("Lab-2 is getting cleaned...\n")

index=index-1

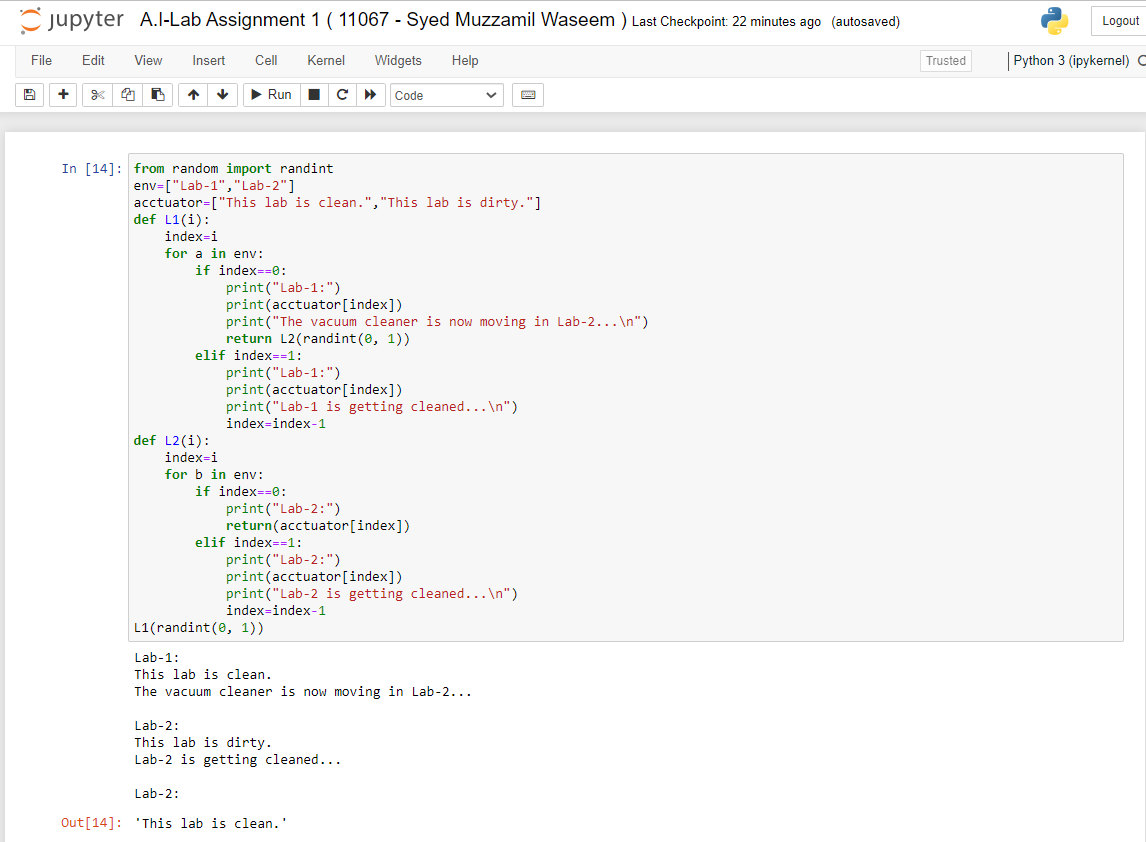
L1(randint(0, 1))

OUTPUT:





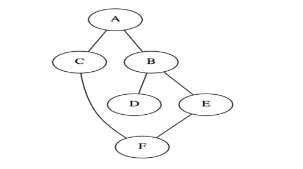




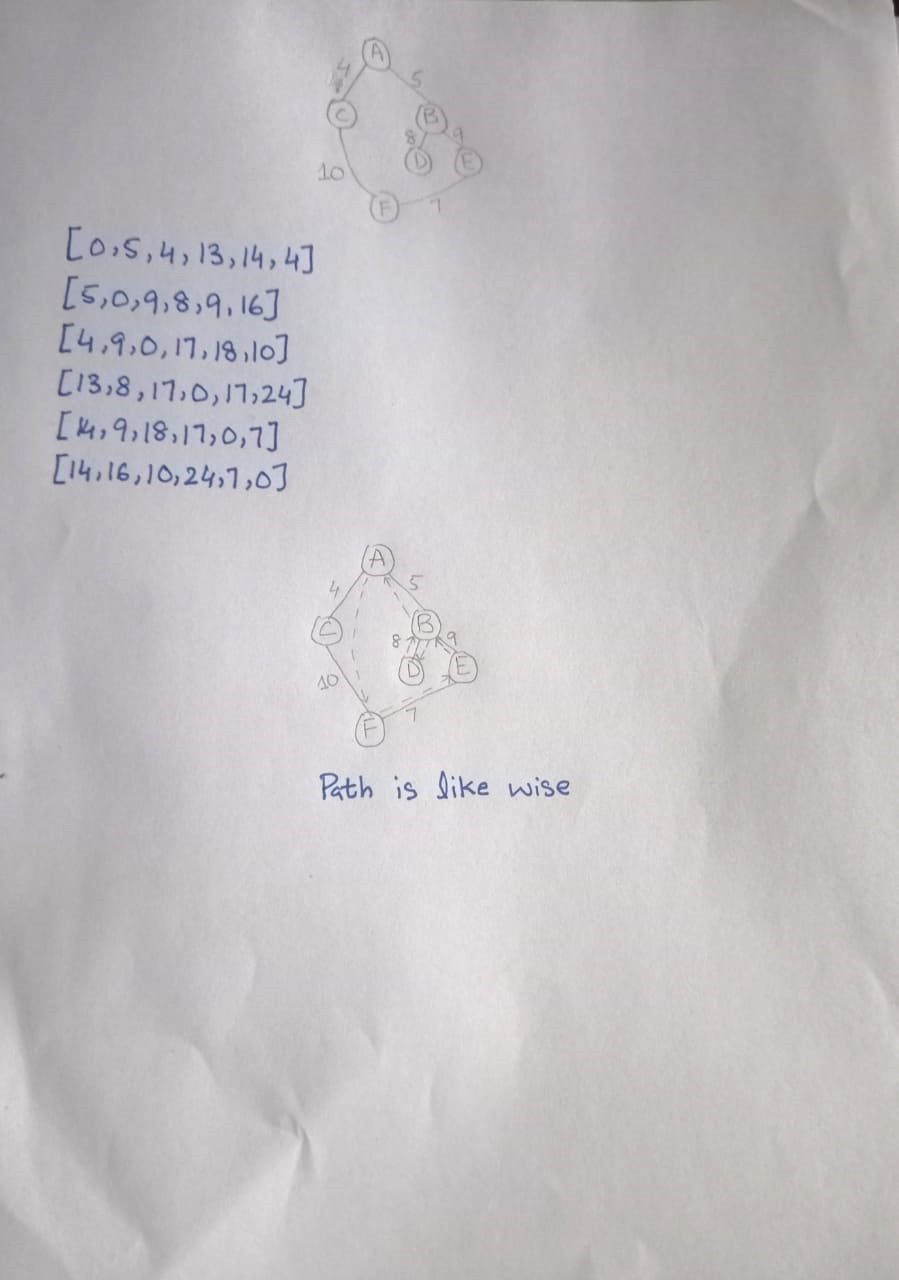
QUESTION 3:

3. Implement the following graph.

## Note: [If Student Id last digit is even perform Travelling Sales Man Problem, If Student Id last digit is odd perform Hill Climbing Algorithm.]



**Done Using TSP and assume the cost by myself**

****

**CODE:**

from itertools import permutations

from sys import maxsize

graph = [[0,5,4,13,14,14], [5,0,9,8,9,16],

[4,9,0,17,18,10], [13,8,17,0,17,24],[14,9,18,17,0,7],[14,16,10,24,7,0]]

a=0

vertex = []

for i in range(6):

if i != a:

vertex.append(i)

minimumpath = maxsize

next\_permutation=permutations(vertex)

# in above line permuatations(vertex) is returning all possible permutations of length vertex

for i in next\_permutation:

current\_pathweight = 0

k = a

for j in i:

current\_pathweight += graph[k][j]

k = j

current\_pathweight += graph[k][a]

minimumpath = min(minimumpath, current\_pathweight)

print("Cost of shortest route is: ",minimumpath)

OUTPUT:

