DATA: 04/01/2022 EXP No: 03	Title of the Lab CSP Implementation	Name:Avinash reddy Vasipalli Registration Number: RA1911027010007 Section: N1 Lab Batch: 1 Day Order: 2

1. Latin Square Problem

AIM:

Implementation of Latin square problem using Python

Concept Description:

Latin square is a n x n grid filled by n distant numbers which are allowed to appear exactly once in a row and a column

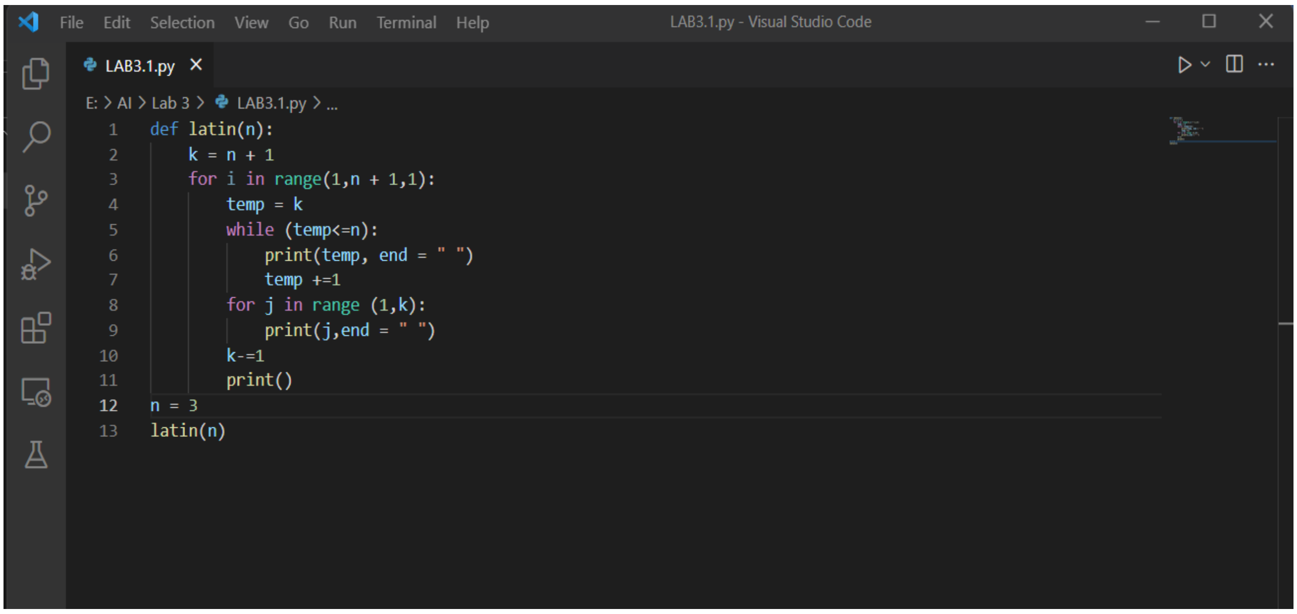
Manual Solution:

Let's understand the solution with an example, In a 3x3 matrix we have to fill 1,2,3 such that they won't repeat in row or column. For this to happen we fill row 1 with 1,2,3. For the next row it has to start with 3 so that left diagonal will have 1.

Finally in third row we again have to start with2 so that matrix can end with 1 again so that the rule is satisfied.

Final output has to be such that each line has 1,2,3 and no 2 numbers are repeated.

Problem Implementation [Coding]:


File Edit Selection View Go Run Terminal Help
LAB3.1.py - Visual Studio Code
LAB3.1.py
E: > AI > Lab 3 > LAB3.1.py > ...

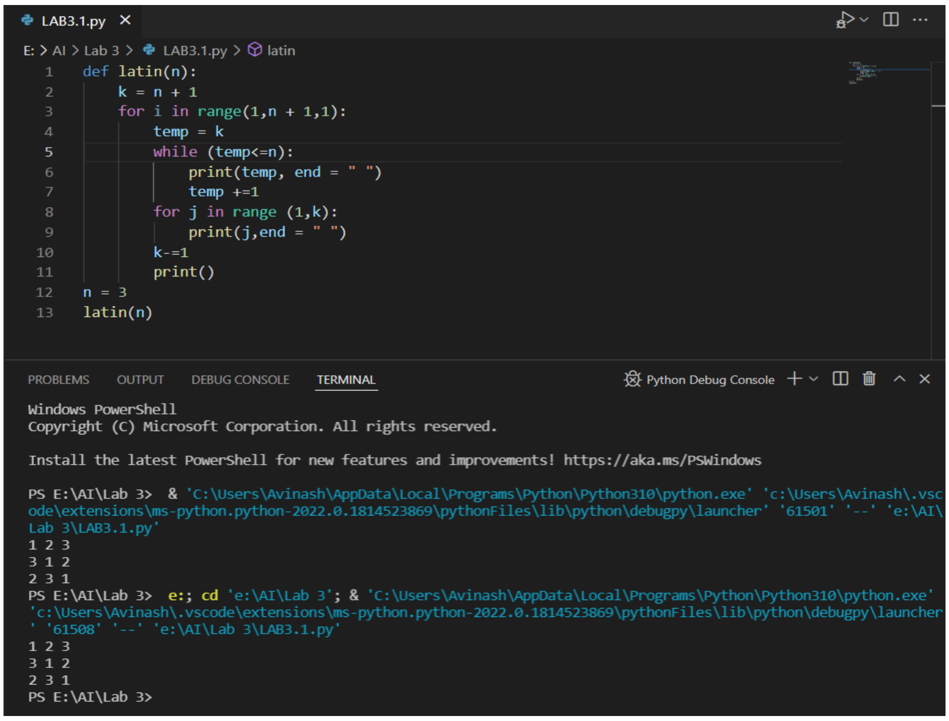
```python
def latin(n):
    k = n + 1
    for i in range(1,n + 1,1):
        temp = k
        while (temp<=n):
            print(temp, end = " ")
            temp +=1
        for j in range (1,k):
            print(j,end = " ")
        k-=1
        print()
n = 3
latin(n)
```

Output:

One such output for 3x3 matrix is **1 2 3**

3 1 2

2 3 1



LAB3.1.py

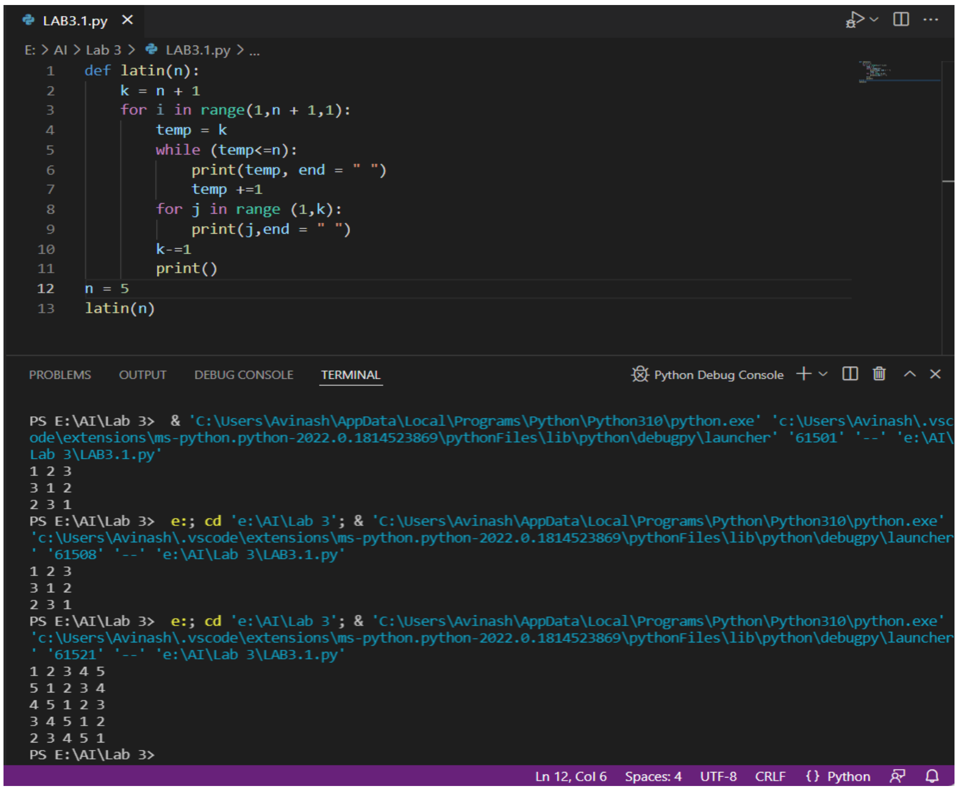
E: > AI > Lab 3 > LAB3.1.py > latin

```python
def latin(n):
    k = n + 1
    for i in range(1,n + 1,1):
        temp = k
        while (temp<=n):
            print(temp, end = " ")
            temp +=1
        for j in range (1,k):
            print(j,end = " ")
        k-=1
        print()
n = 3
latin(n)
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS E:\AI\Lab 3>  & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Avinash\.vsc
ode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher' '61501' '--' 'e:\AI\
Lab 3\LAB3.1.py'
1 2 3
3 1 2
2 3 1
PS E:\AI\Lab 3>  e:; cd 'e:\AI\Lab 3'; & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.exe'
'c:\Users\Avinash\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher
' '61508' '--' 'e:\AI\Lab 3\LAB3.1.py'
1 2 3
3 1 2
2 3 1
PS E:\AI\Lab 3>
```



LAB3.1.py

E: > AI > Lab 3 > LAB3.1.py > ...

```python
def latin(n):
    k = n + 1
    for i in range(1,n + 1,1):
        temp = k
        while (temp<=n):
            print(temp, end = " ")
            temp +=1
        for j in range (1,k):
            print(j,end = " ")
        k-=1
        print()
n = 5
latin(n)
```

```
PS E:\AI\Lab 3>  & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Avinash\.vsc
ode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher' '61501' '--' 'e:\AI\
Lab 3\LAB3.1.py'
1 2 3
3 1 2
2 3 1
PS E:\AI\Lab 3>  e:; cd 'e:\AI\Lab 3'; & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.exe'
'c:\Users\Avinash\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher
' '61508' '--' 'e:\AI\Lab 3\LAB3.1.py'
1 2 3
3 1 2
2 3 1
PS E:\AI\Lab 3>  e:; cd 'e:\AI\Lab 3'; & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.exe'
'c:\Users\Avinash\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\launcher
' '61521' '--' 'e:\AI\Lab 3\LAB3.1.py'
1 2 3 4 5
5 1 2 3 4
4 5 1 2 3
3 4 5 1 2
2 3 4 5 1
PS E:\AI\Lab 3>
```

Ln 12, Col 6 Spaces: 4 UTF-8 CRLF {} Python

Result:

A Latin square problem is solved using python.

2. Room Colouring

Aim:

Implementation of Room colouring problem using Python.

Concept Description:

It is a method of assigning colours to rooms such that all the constraints are satisfied as mentioned in the question or requirements.

Manual Solution:

Set of constraints for the problem is:

- One bedroom can be coloured red only.
- No 2 adjacent rooms can be coloured same.
- Only red, blue, green and violet colours.
- Kitchen preferably coloured blue and never be in green colour.
- Dining room can't be violet

Room Colouring problem as of textbook

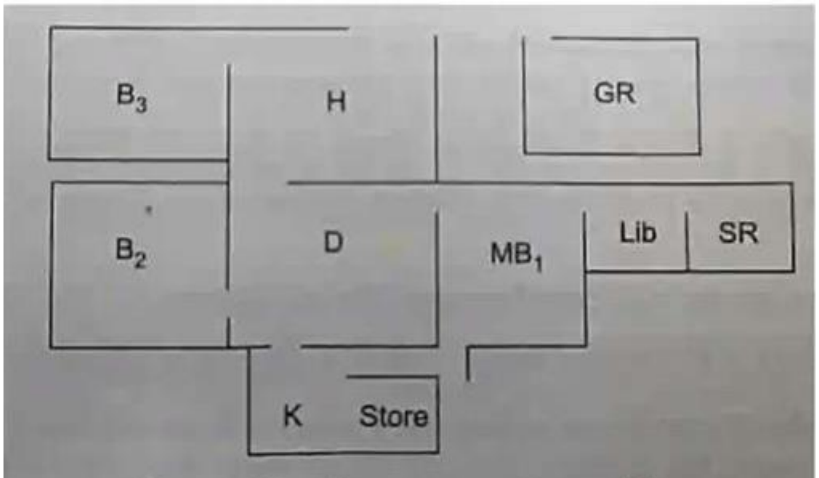
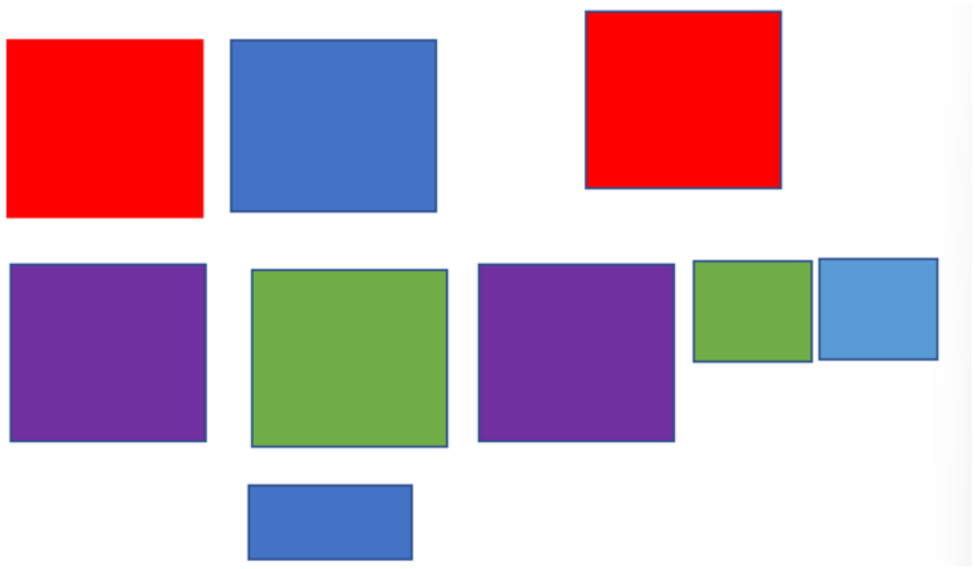
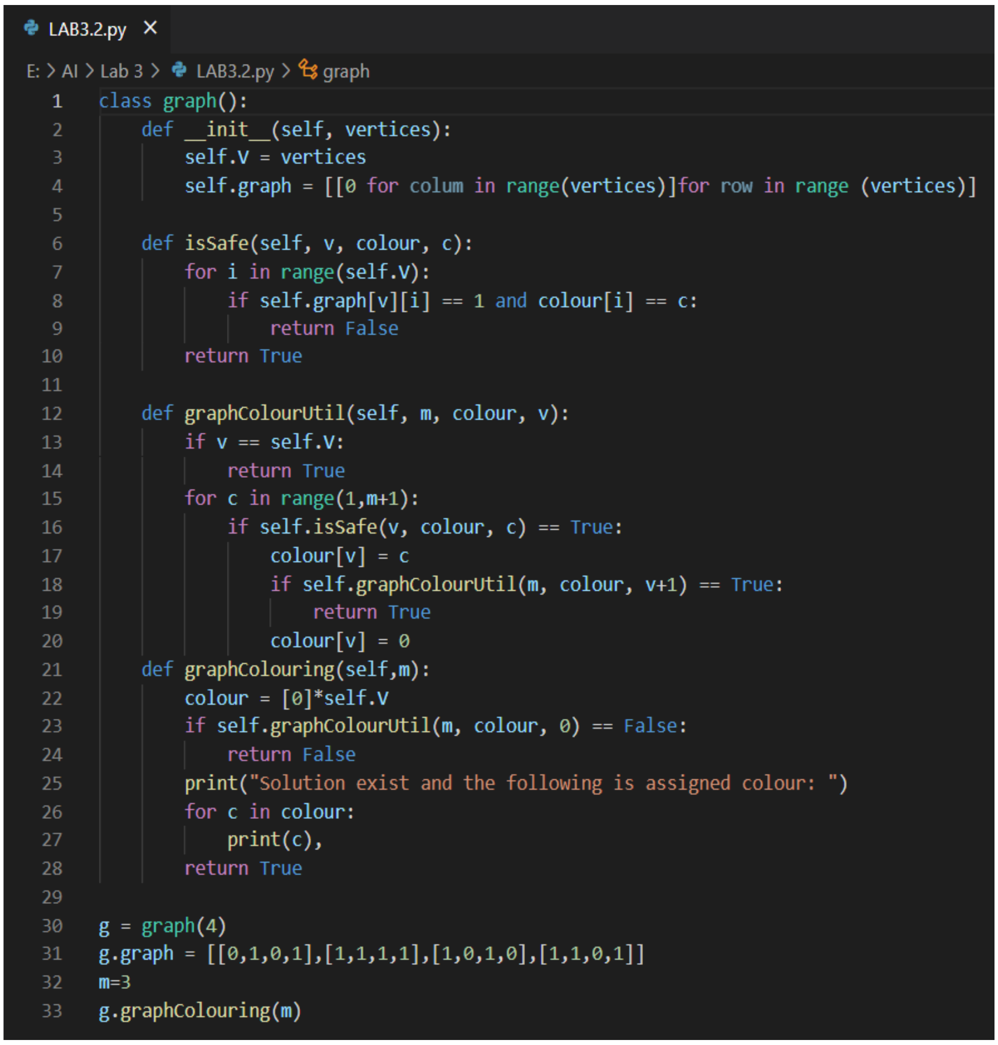


Figure 6.3 Home colouring problem.

With the constraints in mind lets make B3 red. As mentioned, let's make kitchen blue so that one room is reduced. Since B2 is adjacent to B3, kitchen and dining room it can't be red or blue and we know that dining room can't be violet, so let's make B2 violet. Hall can be blue as it doesn't affect any constraints. After all this dining can be green as no rule is mentioned.GR is red and MB is left which can't be blue, green or red as it has adjacent rooms in that colour. Library can be either green or blue. If green then SR is blue and vice versa. The final output is



Program Implementation [Coding]:



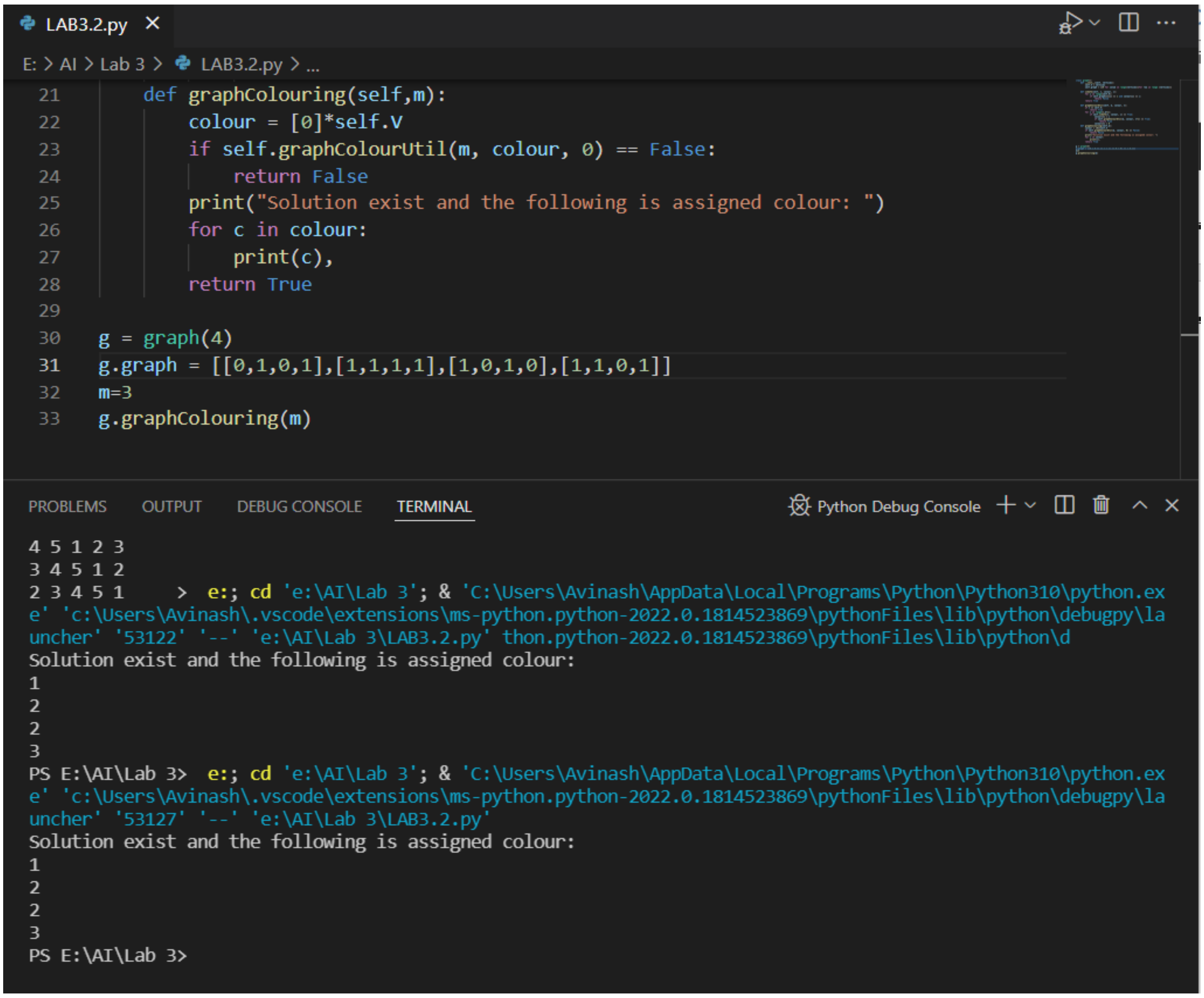
```python
class graph():
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for colum in range(vertices)]for row in range (vertices)]

    def isSafe(self, v, colour, c):
        for i in range(self.V):
            if self.graph[v][i] == 1 and colour[i] == c:
                return False
        return True

    def graphColourUtil(self, m, colour, v):
        if v == self.V:
            return True
        for c in range(1,m+1):
            if self.isSafe(v, colour, c) == True:
                colour[v] = c
                if self.graphColourUtil(m, colour, v+1) == True:
                    return True
                colour[v] = 0
    def graphColouring(self,m):
        colour = [0]*self.V
        if self.graphColourUtil(m, colour, 0) == False:
            return False
        print("Solution exist and the following is assigned colour: ")
        for c in colour:
            print(c),
        return True

g = graph(4)
g.graph = [[0,1,0,1],[1,1,1,1],[1,0,1,0],[1,1,0,1]]
m=3
g.graphColouring(m)
```

Output:



```python
    def graphColouring(self,m):
        colour = [0]*self.V
        if self.graphColourUtil(m, colour, 0) == False:
            return False
        print("Solution exist and the following is assigned colour: ")
        for c in colour:
            print(c),
        return True

g = graph(4)
g.graph = [[0,1,0,1],[1,1,1,1],[1,0,1,0],[1,1,0,1]]
m=3
g.graphColouring(m)
```

```
4 5 1 2 3
3 4 5 1 2
2 3 4 5 1     >  e:; cd 'e:\AI\Lab 3'; & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.ex
e' 'c:\Users\Avinash\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\la
uncher' '53122' '--' 'e:\AI\Lab 3\LAB3.2.py' thon.python-2022.0.1814523869\pythonFiles\lib\python\d
Solution exist and the following is assigned colour:
1
2
2
3
PS E:\AI\Lab 3>  e:; cd 'e:\AI\Lab 3'; & 'C:\Users\Avinash\AppData\Local\Programs\Python\Python310\python.ex
e' 'c:\Users\Avinash\.vscode\extensions\ms-python.python-2022.0.1814523869\pythonFiles\lib\python\debugpy\la
uncher' '53127' '--' 'e:\AI\Lab 3\LAB3.2.py'
Solution exist and the following is assigned colour:
1
2
2
3
PS E:\AI\Lab 3>
```

Result:

Successfully implemented Room colour problem in python

Signature of Student

[illegible]

Avinash reddy Vasipalli