ATMA RAM SANATAN DHARM COLLEGE

Course Title: Discrete Mathematical Structure Practical

Submitted To:

Shalini Ma'am

Faculty Of Computer Science

Submitted By:

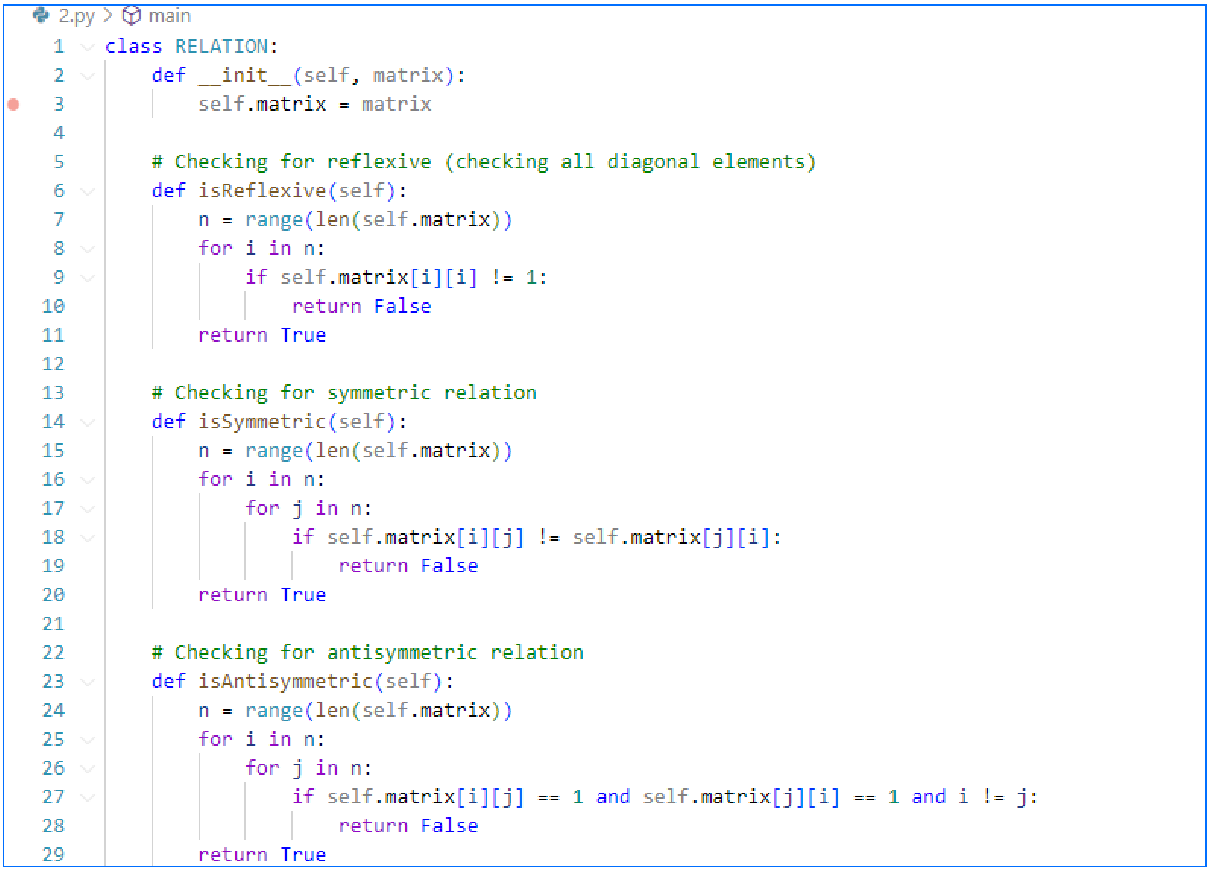
Name	:	Sudeep Kumar Singh
Roll No.	:	22/28021
Course	:	B.Sc. Computer Science Hons.

2. Create a class RELATION, use Matrix notation to represent a relation.

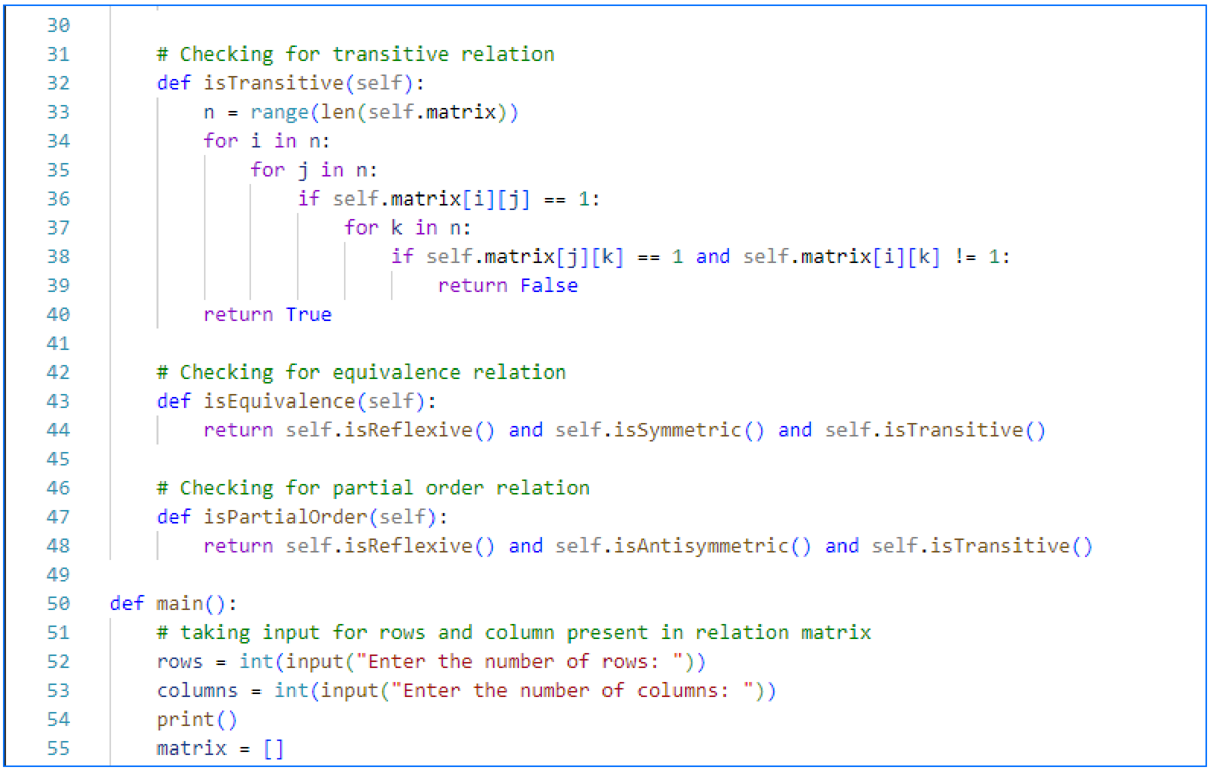
Include member functions to check if the relation is Reflexive, Symmetric, Anti-symmetric, Transitive.

Using these functions check whether the given relation is. Equivalence or Partial Order relation or None.

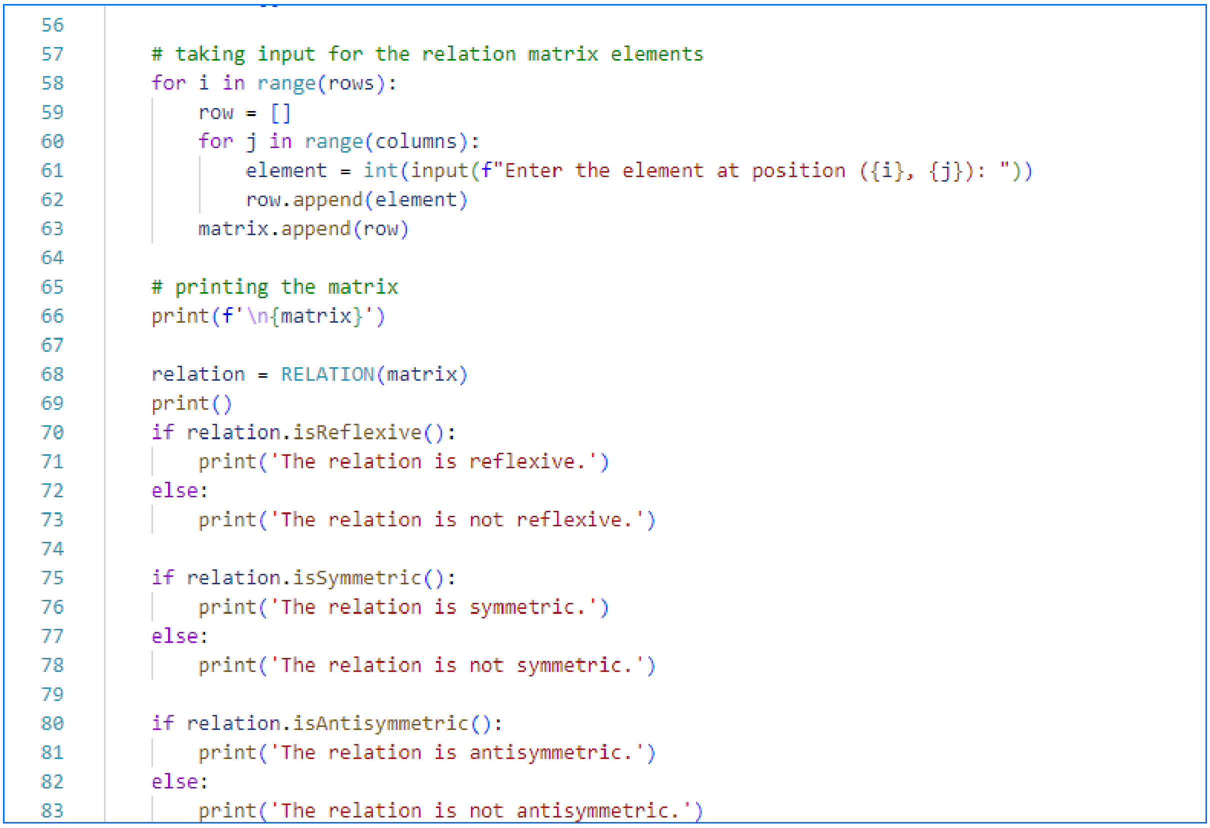
Code:



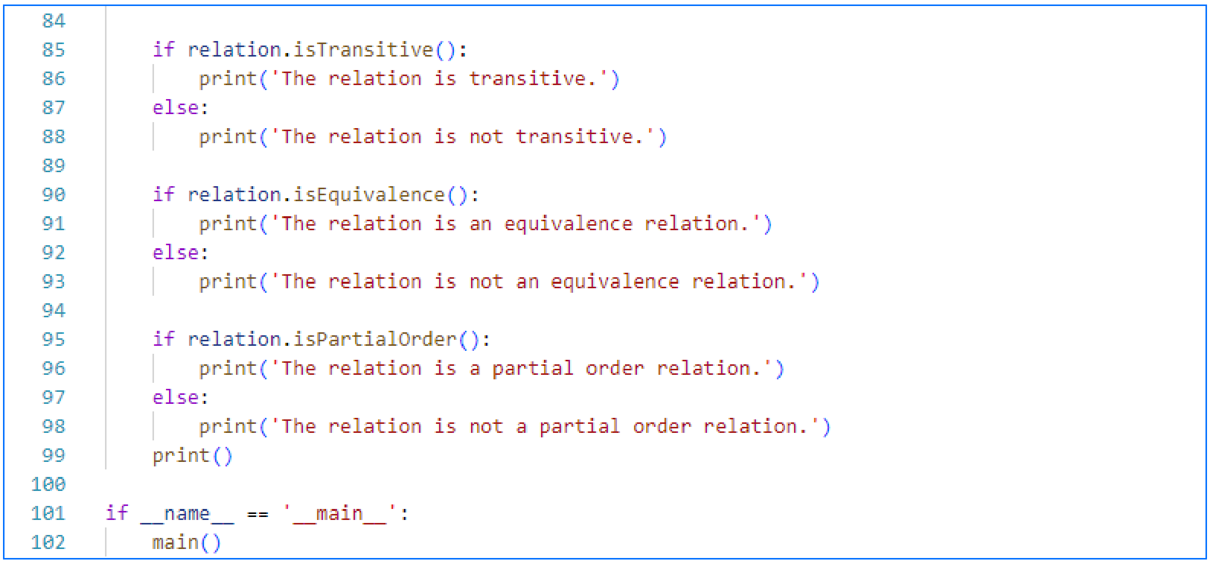
```python
class RELATION:
    def __init__(self, matrix):
        self.matrix = matrix

    # Checking for reflexive (checking all diagonal elements)
    def isReflexive(self):
        n = range(len(self.matrix))
        for i in n:
            if self.matrix[i][i] != 1:
                return False
        return True

    # Checking for symmetric relation
    def isSymmetric(self):
        n = range(len(self.matrix))
        for i in n:
            for j in n:
                if self.matrix[i][j] != self.matrix[j][i]:
                    return False
        return True

    # Checking for antisymmetric relation
    def isAntisymmetric(self):
        n = range(len(self.matrix))
        for i in n:
            for j in n:
                if self.matrix[i][j] == 1 and self.matrix[j][i] == 1 and i != j:
                    return False
        return True
```



```python

    # Checking for transitive relation
    def isTransitive(self):
        n = range(len(self.matrix))
        for i in n:
            for j in n:
                if self.matrix[i][j] == 1:
                    for k in n:
                        if self.matrix[j][k] == 1 and self.matrix[i][k] != 1:
                            return False
        return True

    # Checking for equivalence relation
    def isEquivalence(self):
        return self.isReflexive() and self.isSymmetric() and self.isTransitive()

    # Checking for partial order relation
    def isPartialOrder(self):
        return self.isReflexive() and self.isAntisymmetric() and self.isTransitive()

def main():
    # taking input for rows and column present in relation matrix
    rows = int(input("Enter the number of rows: "))
    columns = int(input("Enter the number of columns: "))
    print()
    matrix = []
```

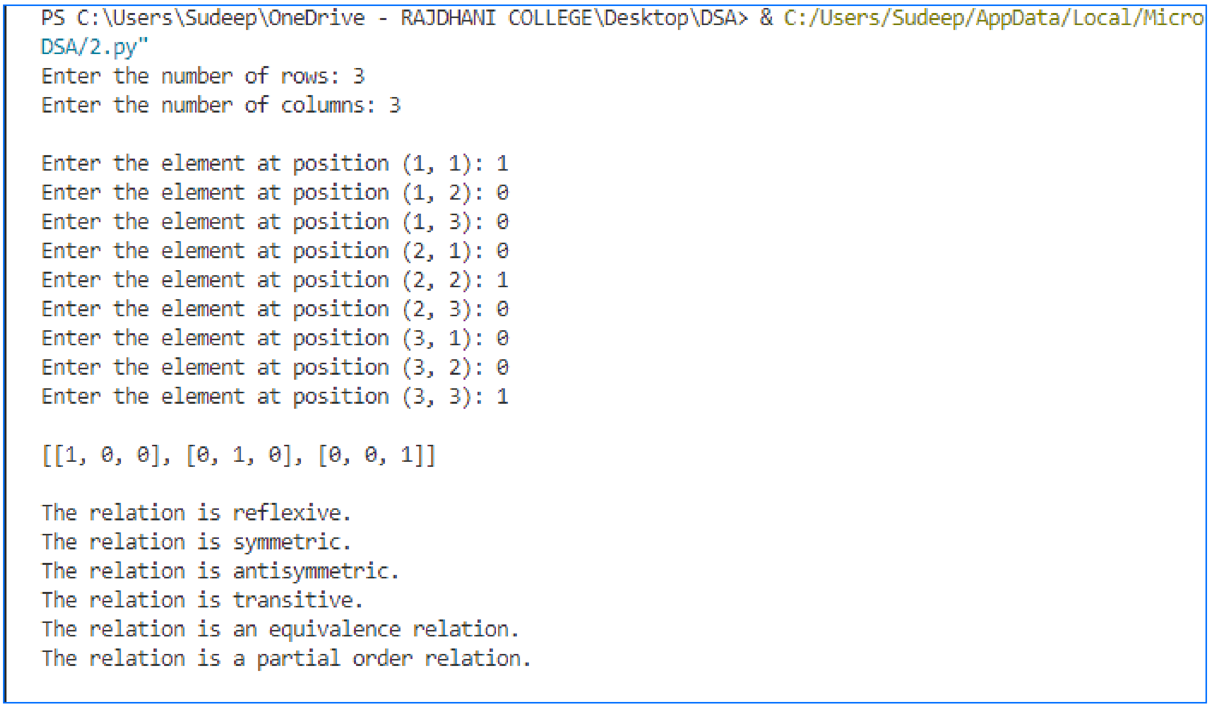


```python

    # taking input for the relation matrix elements
    for i in range(rows):
        row = []
        for j in range(columns):
            element = int(input(f"Enter the element at position ({i}, {j}): "))
            row.append(element)
        matrix.append(row)

    # printing the matrix
    print(f'\n{matrix}')

    relation = RELATION(matrix)
    print()
    if relation.isReflexive():
        print('The relation is reflexive.')
    else:
        print('The relation is not reflexive.')

    if relation.isSymmetric():
        print('The relation is symmetric.')
    else:
        print('The relation is not symmetric.')

    if relation.isAntisymmetric():
        print('The relation is antisymmetric.')
    else:
        print('The relation is not antisymmetric.')
```



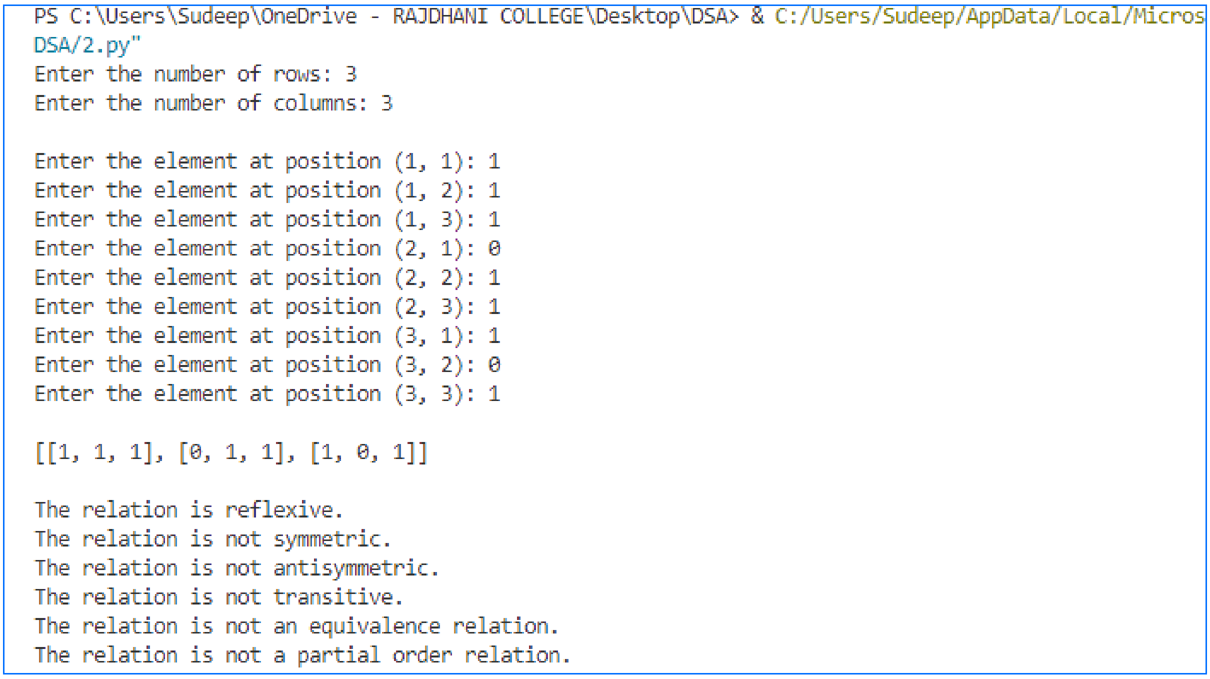
```python

    if relation.isTransitive():
        print('The relation is transitive.')
    else:
        print('The relation is not transitive.')

    if relation.isEquivalence():
        print('The relation is an equivalence relation.')
    else:
        print('The relation is not an equivalence relation.')

    if relation.isPartialOrder():
        print('The relation is a partial order relation.')
    else:
        print('The relation is not a partial order relation.')
    print()

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

Output: 1



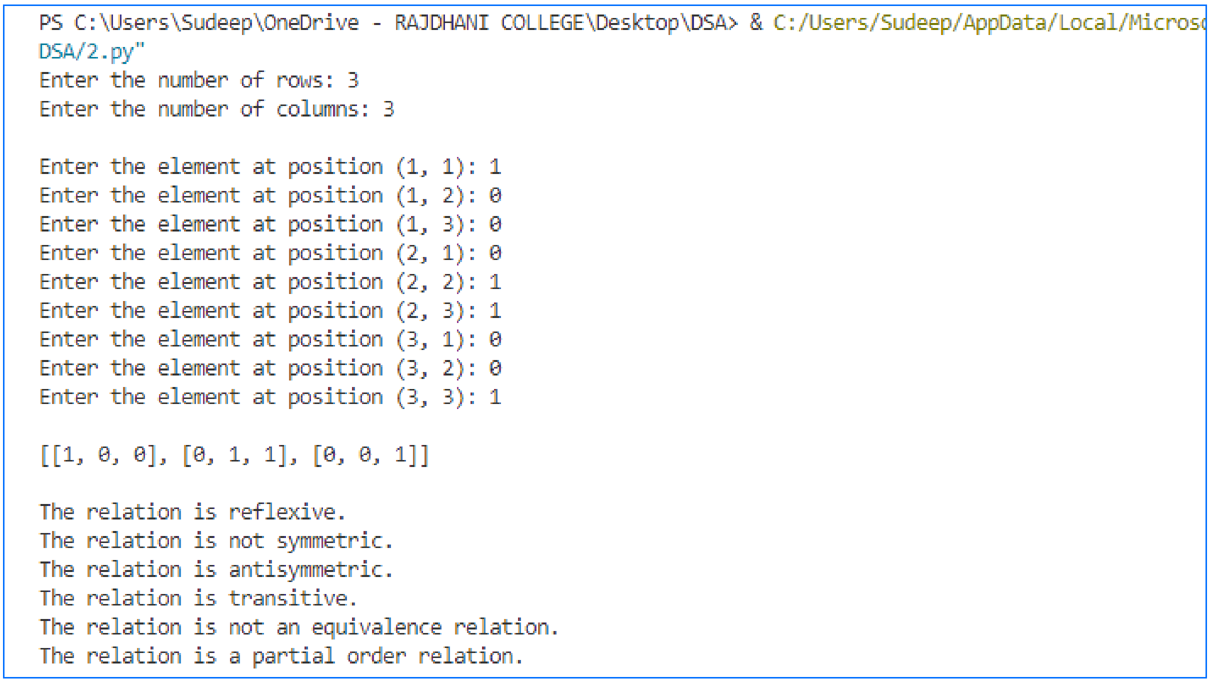
```
PS C:\Users\Sudeep\OneDrive - RAJDHANI COLLEGE\Desktop\DSA> & C:/Users/Sudeep/AppData/Local/Micro
DSA/2.py"
Enter the number of rows: 3
Enter the number of columns: 3

Enter the element at position (1, 1): 1
Enter the element at position (1, 2): 0
Enter the element at position (1, 3): 0
Enter the element at position (2, 1): 0
Enter the element at position (2, 2): 1
Enter the element at position (2, 3): 0
Enter the element at position (3, 1): 0
Enter the element at position (3, 2): 0
Enter the element at position (3, 3): 1

[[1, 0, 0], [0, 1, 0], [0, 0, 1]]

The relation is reflexive.
The relation is symmetric.
The relation is antisymmetric.
The relation is transitive.
The relation is an equivalence relation.
The relation is a partial order relation.
```

2



```
PS C:\Users\Sudeep\OneDrive - RAJDHANI COLLEGE\Desktop\DSA> & C:/Users/Sudeep/AppData/Local/Micros
DSA/2.py"
Enter the number of rows: 3
Enter the number of columns: 3

Enter the element at position (1, 1): 1
Enter the element at position (1, 2): 1
Enter the element at position (1, 3): 1
Enter the element at position (2, 1): 0
Enter the element at position (2, 2): 1
Enter the element at position (2, 3): 1
Enter the element at position (3, 1): 1
Enter the element at position (3, 2): 0
Enter the element at position (3, 3): 1

[[1, 1, 1], [0, 1, 1], [1, 0, 1]]

The relation is reflexive.
The relation is not symmetric.
The relation is not antisymmetric.
The relation is not transitive.
The relation is not an equivalence relation.
The relation is not a partial order relation.
```

3



```
PS C:\Users\Sudeep\OneDrive - RAJDHANI COLLEGE\Desktop\DSA> & C:/Users/Sudeep/AppData/Local/Microso
DSA/2.py"
Enter the number of rows: 3
Enter the number of columns: 3

Enter the element at position (1, 1): 1
Enter the element at position (1, 2): 0
Enter the element at position (1, 3): 0
Enter the element at position (2, 1): 0
Enter the element at position (2, 2): 1
Enter the element at position (2, 3): 1
Enter the element at position (3, 1): 0
Enter the element at position (3, 2): 0
Enter the element at position (3, 3): 1

[[1, 0, 0], [0, 1, 1], [0, 0, 1]]

The relation is reflexive.
The relation is not symmetric.
The relation is antisymmetric.
The relation is transitive.
The relation is not an equivalence relation.
The relation is a partial order relation.
```