Soft Computing - LAB assignment
Name - Madhusudan Chand
Roll No. - B.Sc(Sem-VI)COMP-07

Q1.Implement Fuzzy and Crisp Composition using max-min and max-product mechanisms (Write 4 different files).

Code :

```
#Composition of two crisp relation using max product method
import numpy as np

def maxproduct(x,y):
    z=[]
    for a in x:
        for b in y.T:
            z.append(max(np.multiply(a,b)))
    z=np.array(z).reshape(x.shape[0],y.shape[1])
    print(z)

def takevalue():
    r,c=[int(i) for i in input("Enter the value of row and column of the relation\n").split(' ')]
    return r,c
    pass

row1,col1=takevalue()
print("Values : ")
r1=np.array([[int(input()) for i in range(col1)] for j in range(row1)])
print("Relation 1 : ")
print(r1)

row2,col2=takevalue()
print("Values : ")
r2=np.array([[int(input()) for i in range(col2)] for j in range(row2)])
print("Relation 2 : ")
print(r2)

if(col1==row2):
    print("Composition of two crisp relation using max product method : ")
    maxproduct(r1,r2)
else:
    print("Composition is not done check your matrices..")
```

Output:

```
Enter the value of row and column of the relation
3 4
Values :
1
0
1
0
0
0
0
1
0
0
0
0
Relation 1 :
[[1 0 1 0]
 [0 0 0 1]
 [0 0 0 0]]
Enter the value of row and column of the relation
4 2
Values :
0
1
0
0
0
1
0
0
Relation 2 :
[[0 1]
 [0 0]
 [0 1]
 [0 0]]
Composition of two crisp relation using max product method :
[[0 1]
 [0 0]
 [0 0]]
```

Code :

```
#Composition of two crisp relation using min-max method
import numpy as np
```

```
def maxmin(x,y):
    z=[]
    for a in x:
        for b in y.T:
            z.append(max(np.minimum(a,b)))
    z=np.array(z).reshape(x.shape[0],y.shape[1])
    print(z)

def takevalue():
    r,c=[int(i) for i in input("Enter the value of row and column of the relation\n").split(' ')]
    return r,c
    pass

row1,col1=takevalue()
print("Values : ")
r1=np.array([[int(input()) for i in range(col1)] for j in range(row1)])
print("Relation 1 : ")
print(r1)

row2,col2=takevalue()
print("Values : ")
r2=np.array([[int(input()) for i in range(col2)] for j in range(row2)])
print("Relation 2 : ")
print(r2)

if(col1==row2):
    print("Composition of two crisp relation using min-max method : ")
    maxmin(r1,r2)
else:
    print("Composition is not done check your matrices..")
```

Output:

```
Enter the value of row and column of the relation
3 4
Values :
1
0
1
0
0
0
0
1
0
0
0
```

```
0
Relation 1 :
[[1 0 1 0]
 [0 0 0 1]
 [0 0 0 0]]
Enter the value of row and column of the relation
4 2
Values :
0
1
0
0
0
1
0
0
Relation 2 :
[[0 1]
 [0 0]
 [0 1]
 [0 0]]
Composition of two crisp relation using min-max method :
[[0 1]
 [0 0]
 [0 0]]
```

Code :

```
# Composition of two crisp relation using max product method

import numpy as np

def maxproduct(x,y):
    z=[]
    for a in x:
        for b in y.T:
            z.append(max(np.multiply(a,b)))
    z=np.array(z).reshape(x.shape[0],y.shape[1])
    print(z)
    pass
    pass

def takevalue():
    r,c=[int(i) for i in input("Enter the value of row and column of the relation\n").split(' ')]
    return r,c
    pass
```

```
row1,col1=takevalue()
print("Values : ")
r1=np.array([[float(input()) for i in range(col1)] for j in range(row1)])
print("Relation 1 : ")
print(r1)

row2,col2=takevalue()
print("Values : ")
r2=np.array([[float(input()) for i in range(col2)] for j in range(row2)])
print("Relation 2 : ")
print(r2)

if(col1==row2):
    print("Composition of two crisp relation using max product method : ")
    maxproduct(r1,r2)
else:
    print("Composition is not done check your matrices..")
```

Output :

```
Enter the value of row and column of the relation
2 2
Values :
0.7
0.5
0.8
0.4
Relation 1 :
[[0.7 0.5]
 [0.8 0.4]]
Enter the value of row and column of the relation
2 3
Values :
0.9
0.6
0.2
0.1
0.7
0.5
Relation 2 :
[[0.9 0.6 0.2]
 [0.1 0.7 0.5]]
Composition of two crisp relation using max product method :
[[0.63 0.42 0.25]
 [0.72 0.48 0.2 ]]
```

Code :

```
# Composition of two crisp relation using min-max method
import numpy as np
def maxmin(x,y):
    z=[]
    for a in x:
        for b in y.T:
            z.append(max(np.minimum(a,b)))
    z=np.array(z).reshape(x.shape[0],y.shape[1])
    print(z)
def takevalue():
    r,c=[int(i) for i in input("Enter the value of row and column of the relation\n").split(' ')]
    return r,c
    pass

row1,col1=takevalue()
print("Values : ")
r1=np.array([[float(input()) for i in range(col1)] for j in range(row1)])
print("Relation 1 : ")
print(r1)

row2,col2=takevalue()
print("Values : ")
r2=np.array([[float(input()) for i in range(col2)] for j in range(row2)])
print("Relation 2 : ")
print(r2)

if(col1==row1):
    print("Composition of two crisp relation using min-max method : ")
    maxmin(r1,r2)
else:
    print("Composition is not done check your matrices..")
```

Output :

```
Enter the value of row and column of the relation
2 2
Values :
0.7
0.5
0.8
0.4
Relation 1 :
[[0.7 0.5]
 [0.8 0.4]]
Enter the value of row and column of the relation
```

```
2 3
Values :
0.9
0.6
0.2
0.1
0.7
0.5
Relation 2 :
[[0.9 0.6 0.2]
 [0.1 0.7 0.5]]
Composition of two crisp relation using min-max method :
[[0.7 0.6 0.5]
 [0.8 0.6 0.4]]
```

Q2.Check whether a Fuzzy relation satisfies the equivalence property or not, if no, then display the reason also

Code:

```
import numpy as np
flag1=False
flag2=False
flag3=False
def takevalue():
   r,c=[int(i) for i in input("Enter the value of row and column of the relation\n").split(' ')]
   return r,c
   pass

row , col=takevalue()

if(row != col):
   print("relation must be a square matrix.");
else:
   print("Values:")
   r=np.array([[float(input()) for i in range(col)] for j in range(row)])

   def minimum(a,b):
         if(a>b):return b
      else:return a
      pass
   def checkreflexive():
      for i in range(row):
         for j in range(col):
                  if(r[i][i]!=1.0):
```

```
                print("Relation is not reflexive")
                return False
        return True
        pass
    def checksymmetric():
        for i in range(row):
            for j in range(col):
                if(r[i][j]!=r[j][i]):
                    print("Relation is not symmetric")
                    return False
        return True
    def checktransitive():
        for i in range(row):
            for j in range(col):
                for k in range(row):
                    if((r[i][k]>=minimum(r[i][j],r[j][k]))):
                        return True
        print("Relation is not transitive")
        return False
    if(checkreflexive() and checksymmetric() and checktransitive()):

        print("Fuzzy relation is equivalence..")

    else:
        print("Fuzzy relation is not equivalence...")
```

Output :

```
Enter the value of row and column of the relation
2 2
Values:
1.0
0
1.0
0
Relation is not reflexive
Fuzzy relation is not equivalence...
```

Dated : 04/05/2023

Output :

```
Enter the value of row and column of the relation
2 2
Values:
1.0
0
0
1.0
Fuzzy relation is equivalence..
```