== ==

1. **Write a program in MATLAB to perform Union, Intersection and Complement operations.**

```
%Enter Data
u=input ('Enter First Matrix');
v=input ('Enter Second Matrix');
%To Perform Operations
w=max (u, v); p=min (u, v); q1=1-u; q2=1-v;
%Display Output
display ('Union of Two Matrices');
 display(w); display ('Intersection of Two Matrices');
 display(p); display ('Complement of First Matrix');
 display(q1); display ('Complement of Second Matrix');
display(q2); OUTPUT: -Enter First Matrix [0.2,0.4]
Enter Second Matrix [0.3,0.5]
Union of Two Matrices w = 0.30000.5000
Intersection of Two Matrices
p = 0.20000.4000
Complement of First Matrix q1 = 0.80000.6000
Complement of Second Matrix q2 = 0.70000.5000
```

== ==

```
clc
 a= [ 0.1 0.5 0.9 0.3]
 b= [ 0.9 0.3 0.8 0.1]
disp ('=========================================');
disp ('Demorgans Law');
demu= (1-max (a, b))
 demu1= min(1-a,1-b)
 if(demu==demu1)
        disp ('Demorgans Law is Satisfied Union')
 end
disp ('=========================================='); demi= (1-
min (a, b))
demi1=min(1-a,1-b)
 if(demi==demi1)
         disp ('Demorgans Law is Satisfied for Intersection')
 end
```

==

==

2. Write a program in MATLAB to implement De-Morgan's Law

```
%Enter Data

u=input ('Enter First Matrix');

v=input ('Enter Second Matrix');

%To Perform Operations

w=max (u, v);
p=min (u, v);
q1=1-u;
q2=1-v;
x1=1-w;
x2=min (q1, q2);
y1=1-p;
y2=max (q1, q2);

%Display Output

display ('Union of Two Matrices');

display (w);
display ('Intersection of Two Matrices');
display (p);
display ('Complement of First Matrix');
display(q1);
display ('Complement of Second Matrix');
display (q2);
display ('De-Morgan's Law');
display ('LHS');
display (x1);
display ('RHS');
display (x2);
display ('LHS1');
display (y1);
display ('RHS1');
display (y2);
```

OUTPUT: - Enter First Matrix [0.3,0.4] Enter Second Matrix [0.9,0.8]

Union of Two Matrices
3w =0.9000 0.8000
Intersection of Two Matrices p =0.3000 0.4000
Complement of First Matrix q1 =0.7000 0.6000
Complement of Second Matrix q2 =0.1000 0.2000
De-Morgan's Law LHS x1 =0.1000 0.2000
RHS x2 =0.1000 0.2000 LHS1
4y1 =0.7000 0.6000RHS1y2 =0.7000 0.6000

== ==

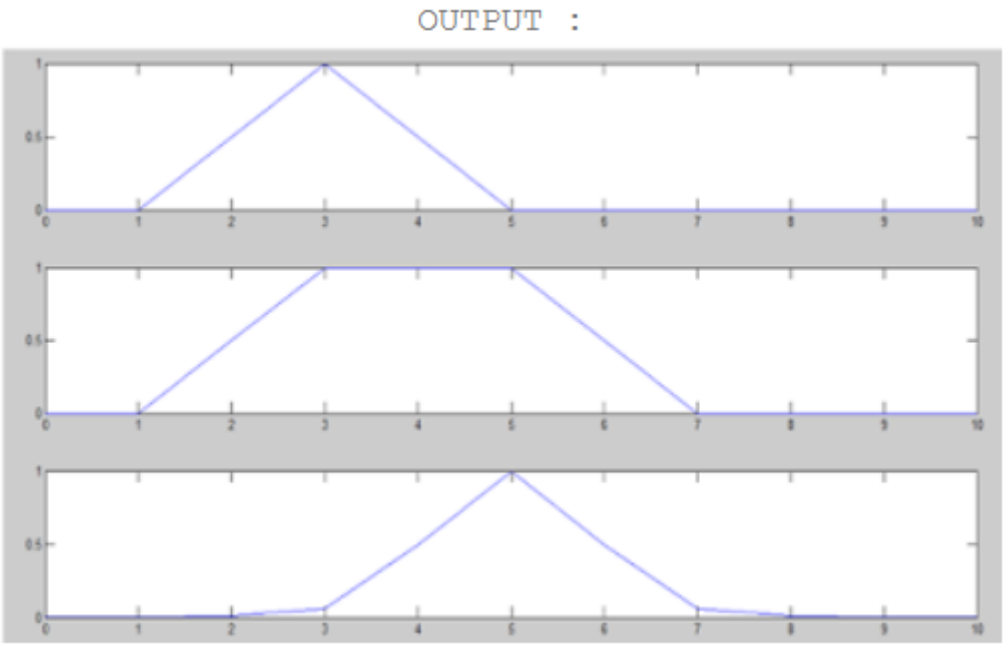
3. To plot various membership functions (triangular, trapezoidal and bell shaped)

```
%Triangular Membership function
x= (0.0:1.0:10.0)';
y1=trimf (x, [1 3 5]);
subplot (311)
plot (x, [y1]);

%Trapezoidal Membership function
x= (0.0:1.0:10.0)';
y1=trapmf (x, [1 3 5 7]);
subplot (312)
plot (x, [y1]);

%Bell Shaped Membership function
y1=gbellmf (x, [1 2 5])';
subplot (313)
plot (x, [y1]);
```

OUTPUT:



== ==

4. **Use Fuzzy toolbox to model tip value that is given after a dinner which can be-not good, satisfying, good and delightful and service which is poor, average or good and the tip value will range from Rs. 10 to 100.**

We are given the linguistic variables quality of food and service as input variables which can be written as: Quality (not good, satisfying, good, delightful) Service (poor, average, good) Similarly Output variable is Tip_value which may range from Rs. 10 to 100.

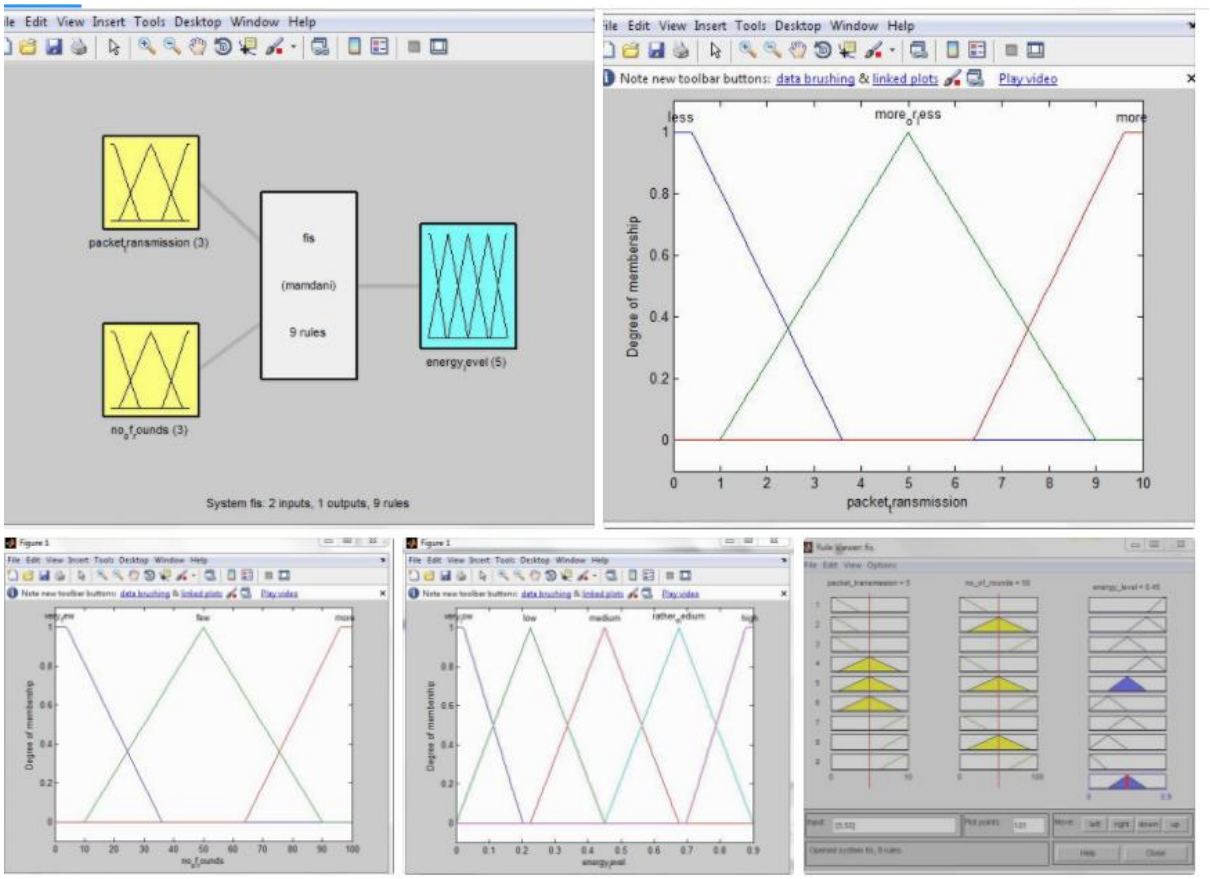
A Fuzzy system comprises the following modules: -

1. Fuzzification Interface

2. Fuzzy Inference Engine

3. Defuzzification

Interface Fuzzy sets are defined on each of the universe of discourse: -Quality, service and tip value.



```
a=newfis('tipper');
a=addvar (a,'input','service’, [0 10]);
a=addmf (a,'input',1,'poor','gaussmf’, [1.5 0]);
a=addmf (a,'input',1,'good','gaussmf’, [1.5 5]);
a=addmf (a,'input',1,'excellent','gaussmf’, [1.5 10]);
a=addvar (a, 'input', 'food’, [0 10]);
a=addmf (a,'input',2,'rancid','trapmf’, [-2 0 1 3]);
a=addmf (a,'input',2,'delicious','trapmf’, [7 9 10 12]);
a=addvar (a, 'output', 'tip’, [0 30]);
a=addmf (a,'output',1,'cheap','trimf’, [0 5 10]);
a=addmf (a,'output',1,'average','trimf’, [10 15 20]);
a=addmf (a,'output',1,'generous','trimf’, [20 25 30]);
ruleList= [ ...
1 1 1 1 2
2 0 2 1 1
3 2 3 1 2];
a=addrule (a, ruleList);
showfis(a);
showrule(a);
output = evalfis ([ 1 2], a); % Output calculation, input = [ 1 2]
```

==

==

5. Generate AND, NOT function using McCulloch-Pitts neural net by MATLAB

```
%Getting weights and threshold valuedisp ('Enter weights');

w1=input ('Weight w1=');

w2=input ('weight w2=');
disp ('Enter Threshold Value');

theta=input('theta=');

y= [0 0 0 0];
x1= [0 0 1 1];
x2= [0 1 0 1];
z= [0 0 1 0];

con=1;

while con zin=x1* w1+ x2 * w2;

for i=1:4
if
        zin(i) >= theta y(i)=1;

else
        y(i)=0;

end
end

disp ('Output of Net');

disp(y);

if y==z con=0;
else
disp ('Net is not learning enter another set of weights and Threshold value');

w1=input ('weight w1=');

w2=input ('weight w2=');

theta=input('theta=');
end
end
disp ('McCulloh-Pitts Net for ANDNOT function');
disp ('Weights of Neuron');
disp(w1);
disp(w2);
disp ('Threshold value');
disp(theta);
```

== ==

6. Generate XOR function using McCulloch-Pitts neuron.

The truth table for the XOR function is:

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

The MATLAB program is given by CODE:

```
%XOR function using McCulloch-Pitts neuron

%Getting weights and threshold value

disp ('Enter weights');

w11=input ('Weight w11=');

w12=input ('weight w12=');

w21=input ('Weight w21=');

w22=input ('weight w22=');

v1=input ('weight v1=');

v2=input ('weight v2=');

disp ('Enter Threshold Value');

theta=input('theta=');
x1= [0 0 1 1];
x2= [0 1 0 1];
z= [0 1 1 0];
con=1;
while con zin1=x1 * w11 + x2 * w21;

zin2=x1*w21+x2*w22;

for i=1:4
if
 zin1(i)>=thetay1(i)=1;

else y1(i)=0;

end if
 zin2(i)>=thetay2(i)=1;

else y2(i)=0;

end
end

yin=y1*v1+y2*v2;
for i=1:4
if yin(i)>=theta;
y(i)=1;
else
y(i)=0;
end
```

```
end
disp ('Output of Net');
disp(y);

if y==z

con=0;
else

disp ('Net is not learning enter another set of weights and Threshold value')

w11=input ('Weight w11=');

w12=input ('weight w12=');

w21=input ('Weight w21=');

w22=input ('weight w22=');

v1=input ('weight v1=');

v2=input ('weight v2=');

theta=input('theta=');

end

end

disp ('McCulloch-Pitts Net for XOR function');

disp ('Weights of Neuron Z1');

disp(w11);

disp(w21);

disp ('weights of Neuron Z2');

disp(w12);

disp(w22);

disp ('weights of Neuron Y');

disp(v1);

disp(v2);

disp ('Threshold value');

disp(theta);
```