X1	X2	Y
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

```
clear all; clc;
disp("ADALINE NETWORK FOR OR FUNCTION BIPOLAR INPUTS AND TARGET");
i1 = [1 1 -1 -1]; i2 = [1 -1 1 -1]; %bias input i3 = [1 1 1 1]; %target vector t  = [1 1 1 -1];
% Assigning initial networks weights and bias
w1 = 0.1; w2 = 0.1; b = 0.1;
%First initializing  the learning rate
alpha = 0.1;
%error convergence e
= 0;
%change in weights and bias
delwl = 0; delw2 = 0;
delb = 0; epoch = 0;
while(e < 0.5)
epoch = epoch + 1;
e = 0;     for j = 1:4
       finaly(j) = w1 * i1(j) + w2 * i2(j) + b;
%Inet input calculated and targeted
```

```
        nt = [finaly(j) t(j)];          delwl = alpha
* (t(j) - finaly(j)) * i1(j);          delw2 = alpha
* (t(j) - finaly(j)) * i2(j);
        delb=alpha * (t(j) - finaly(j)) * i3(j);
%Weight changes          wc = [delwl
delw2 delb];          %updation of
weights          wl = w1+delwl;          w2
= w2+delw2;          b = b + delb;
%new weights          w = [wl w2 b];
%input pattern
        i = [i1(j) i2(j) i3(j)];          %now
printing output          out = [i nt wc w]
end     for k=1:4          finaly(k) = w1 *
i1(k) + w2 * i2(k) + k          e=e + (t(k) -
finaly(k)) ^ 2;     end     if epoch == 1
end
end    end
    for i = 1:4          nety(i) = w1 * x1(i) +
w2 * x2(i) + b;          e = e + (t(i) - nety(i))
^ 2;
    end end
```