```c
/*Training a Rosenblatt's Perceptron to learn a
Linearly separable function- AND, OR,NAND and NOR Gates.
Author : Srinath Krishnamoorthy
Date   : 19-June-2017 */

#include <stdio.h>
#include <stdlib.h>
int main()
{
    int x[4][3],i,j,Ya[4],Yd[4],err,epoch;
    float net,lr,w[3];
    /* x[][] is the input patern where X0 is the Bias and always set to 1
    i,j are used for the matrix manipulation- rows and columns respectively
    Ya[] is the actual output of the perceptron after calculating the
    aggregated sum of weights and their respective inputs.
    If the net value;given bt varable 'net' is above a threshold
    (here 0) the value of Ya[] is 1 else 0.
    err- is the variable that capures the error= Yd-Ya;
    where Yd is the expected or desired output
    w[] - gives the wieghts associated with each input to the neuron
    lr - is the learning coefficient such that 0<lr<1
    epoch - is the number of iterations on all the test data till we get zero error*/

    printf("*****Training a Perceptron****** \n ");
```

```c
/*Read the input matrix - 4x3.
Here X0 is the Bias - always set to 1
    X1 and X2 range from 00-11 */

printf("Enter the inputs - \n");
for(i=0;i<4;i++)
{
    for(j=0;j<3;j++)
    {
        scanf("%d",&x[i][j]);
    }
}

/* Now input the initial weights associated with each input */
printf("Enter the weights associated with each input-\n");
for(i=0;i<3;i++)
{
    scanf("%f",&w[i]);
}

/*Desired pattern that you wish the perceptron to learn.
  AND - Will have Yd=[0 0 0 1]
  OR - Will have Yd=[0 1 1 1]
  Any linearly separable input is fine.
  Just try the Yd for XOR. What do you notice? */
```

```
printf("Enter the desired output - \n");
for(i=0;i<4;i++)
{
    scanf("%d",&Yd[i]);
}

// 0<lr<1.
printf("Enter the learning coefficient - \n");
scanf("%f",&lr);

printf("\n");

//print inputs

printf("Bias \t X1 \t X2 \n");
for(i=0;i<4;i++)
{
    for(j=0;j<3;j++)
    {
        printf("%d \t ",x[i][j]);
    }
    printf("\n");
}
printf("\n\n\n");
```

```
//print weights
printf("W0 \t W1 \t W2 \n");
for(i=0;i<3;i++)
{
    printf("%.2f \t ",w[i]);
}
printf("\n\n\n");

//print Yd
printf("Desired output-Yd \n");
for(i=0;i<4;i++)
{
    printf("%d \n",Yd[i]);
}

printf("\n\n");

printf("%.2f is the learning coefficient \n\n", lr);

err=0; //Yd[x]-Ya[x]
epoch=0; // iterations set to 0
net=0.00; // WiXi-cumulative sum initialised to 0

printf("W0 \t W1 \t W2 \t NET OUTPUT \t Ya \t Yd \n\n");
```

```
do
{
    for(i=0;i<4;i++)
    {
        for(j=0;j<3;j++)
        {
            net=net+(w[j]*x[i][j]);
        }

        /* 0 is the threshold for the perceptron to fire.
        Here we use a step function for activation */
        if(net>=0)
        {
            Ya[i]=1;
        }
        else
        {
            Ya[i]=0;
        }

        //error is difference between desired and expected output
        err=Yd[i]-Ya[i];
```

```
            //updating the weights for the current epoch
            for(j=0;j<3;j++)
            {
                w[j]=w[j]+(lr*x[i][j]*err);
                printf("%.2f \t",w[j]);
            }

            printf("%.2f \t\t  %d \t %d \n",net,Ya[i],Yd[i]);

        }
        epoch++;

    }while(Ya[0] != Yd[0] || Ya[1] != Yd[1] || Ya[2] != Yd[2] || Ya[3] != Yd[3]);

    /*Execute the above do-while loop till Yd[] = Ya[]*/

    printf("\nFor learning coefficient %.2f number of epochs is %d \n",lr,epoch);

    return 0;
}
```