**PREGUNTA 1 SISTEMA EXPERTO DE LA MONEDA:**

// #include <iosfwd>

#include <stdio.h>

#include <conio.h>

//#include <iostream>

void interfaz(void);

int inferencia(char \_diametro, char \_color, char \_decoracion);

//diametro

//color

//decoracion

void reportar(int \_accion);

char diametro,color,decoracion;

void main()

{ int accion;

 interfaz();

 accion = inferencia(diametro, color, decoracion);

 reportar(accion);

}

void interfaz(void)

{

cout<<"-- Sistema de Monedas 1, 2 y 5 soles --"<<endl;

cout<<"\n\n"<<"D"<<endl;

cout<<"\n\n"<<"a. 20mm\nb. 23mm\nc. 25mm"<<endl;>>diametro;

cout<<"\n\n"<<"COLOR"<<endl;

cout<<"\n\n"<<"a. Plata\nb. Plata y Cobre\nc. Cobre "<<endl; cin>>color;

cout<<"\n\n"<<"DECORACION"<<endl;

cout<<"\n\n"<<"a. Ave\nb. Lineas de Nazca\nc. Palma"<<endl; cin>>decoracion;

}

int inferencia(char \_diametro, char \_color, char \_decoracion)

{ if(\_diametro=='c' && \_color=='a' && \_decoracion=='c'){

 return(0); // Moneda de 1 sol

 }

 if(\_diametro=='a' && \_color=='b' && \_decoracion=='b'){

 return(1); // Moneda de 2 soles

 }

 if(\_diametro=='b' && \_color=='c' && \_decoracion=='a'){

 return(2); // Moneda de 5 soles

 }

 else

 {

 return(3); // Si en caso se equivocan

 }

}

void reportar(int \_accion)

{

 if(\_accion==0)

 {

 cout<<"\n\n\t\t"<<"--------- MONEDA DE 1 SOL ---------"<<endl;

 getch();

 }

 if(\_accion==1)

 {

 cout<<"\n\n\t\t"<<"--------- MONEDA DE 2 SOL ---------"<<endl;

 getch();

 }

 if(\_accion==2)

 {

 cout<<"\n\n\t\t"<<"--------- MONEDA DE 3 SOL ---------"<<endl;

 getch();

 }

 if(\_accion==3)

 {

 cout<<"\n\n\t\t"<<"--------- VOLVER A INTENTAR ---------"<<endl;

 getch();

 }

}

**PREGUNTA 3**

>> MIRED = newff([0 1; 0 1],[2 1],{'logsig','logsig'})

 MIRED=



>> input = [0 0 1 1; 0 1 0 1];

>> input = [0 0 1 1; 0 1 0 1]



>> output=sim(MIRED,input)



>> target = [0 1 1 0]



>> plot(target, 'o')



>> hold on

>> plot(output, '+r')

>> MIRED.IW{1,1}



>> net.IW{1,1}(1,2)=5;

>> net.IW{1,1}

ans =

 7.7055 5.0000

 -7.9089 -0.4123

>> net.LW{2,1}

ans =

 4.6527 3.1164

>> output=sim(net,input)

output =

 0.1025 0.0846 0.0659 0.6645

>> plot(output,'g\*')

>> net = train(net,input,target);

TRAINLM, Epoch 0/100, MSE 0.540649/0, Gradient 0.164636/1e-010

TRAINLM, Epoch 19/100, MSE 0.125/0, Gradient 3.63247e-011/1e-010

TRAINLM, Minimum gradient reached, performance goal was not met.

>> output = sim(net,input)

output =

 0.0000 1.0000 0.5000 0.5000

>> net.IW{1,1}

ans =

 33.7087 7.4213

 -33.7592 7.1341

>> net.LW{2,1}

ans =

 13.7306 14.8288

>>

**pregunta 4:**



>> x=linspace(1,100); % dominio de 1 a 100

>> y=rand(1,100); % números aleatorios en el intervalo de 0 a 1

>> generaciones = 800;

>> n = 200;

>> probabilidad = 0.6;

>> mutacion = [1 1];

>> poblacion = rand(n,2);

>> s = n;

>> hw = waitbar(0,'Evolucionando..MI SISTEMA.');

>> for i = 1:generaciones

poblacion(:,3) = rand(s,1);

poblacion = sortrows(poblacion,+3);

poblacion=[poblacion;

poblacion(1:2:end-1,1) poblacion(2:2:end,2) zeros(fix(s/2),1);

poblacion(2:2:end,1) poblacion(1:2:end-1,2) zeros(fix(s/2),1)];

s = size(poblacion,1);

poblacion=[poblacion;

poblacion+[(rand(s,3)<probabilidad).\*(rand(s,3)\*2-1).\*repmat([mutacion 0],s,1)]];

s = size(poblacion,1);

**//valor de los parmetros**

m = repmat(poblacion(:,1),1,100);

b = repmat(poblacion(:,2),1,100);

X = repmat(x,s,1);

Y = repmat(y,s,1);

Y2=m.\*X+b;

 poblacion(:,3)=sum(abs(Y-Y2),2);

 poblacion = unique(poblacion,'rows');

 poblacion = sortrows(poblacion,+3);

s = min([size(poblacion,1),n]);

poblacion = poblacion(1:s,:);

 waitbar(i/generaciones,hw)

end

>> close(hw)

>> error\_ag = poblacion(1,3);

>> p = polyfit(x,y,1);

>> y3 = polyval(p,x);

>> error\_mc = sum(abs(y3-y));

>> plot(x,y,'.',x,polyval(poblacion(1,1:2),x),x,y3,':')

>> axis([0 100 -0.5 1.5])

>> legend('DS','AG','MC')

>> text(1,-.167,['AG: Error = ' num2str(error\_ag) ', m = ' num2str(poblacion(1,1)) ', b = '

num2str(poblacion(1,2))])

>> text(1,-.167,['AG: Error = ' num2str(error\_ag) ', m = ' num2str(poblacion(1,1)) ', b = ' num2str(poblacion(1,2))])

>> text(1,-.333,['MC: Error = ' num2str(error\_mc) ', m = ' num2str(p(1)) ', b = ' num2str(p(2))])

>>

