```
void mostrar(int a[][n],int tl){

    for(int i=0;i<tl;i++){
        for(int j=0;j<tl;j++){
            cout<<a[i][j]<<" ";}
        cout<<endl;}
}
//-----------------------------------------------

bool esNula(int a[][n],int tl){
    int i=-1;     bool nula=true;
        while(nula and i<tl){         int j=0; i++;
                while(nula and j<tl){(a[i][j]!=0)? nula=false: j++;}}
    return nula;}
//-----------------------------------------------------
bool simetrica(int a[][n],int tl){
    int i=0,sim=1;
while(i<tl and sim){
    int j=0;
    while(j<tl and a[i][j]==a[j][i]) j++;
    if(j<tl){sim=0;}
    i++;}
return sim;}
//-----------------------------------------------------
bool DiagonalNula(int a[][n],int tl){
    int i=0;
    while(i<tl and a[i][i]==0)i++;
    return(i<tl)? false:true;}
//-----------------------------------------------------
bool TriangularSuperior(int a[][n],int tl){ //elementos por debajo de
la diagonal principal nulos
    int i=1,triangular=1;
    while(i<tl and triangular){    int j=0;
    while(j<i and a[i][j]==0) j++;
    (j<i)? triangular=0:i++;}
        return triangular;}
//-----------------------------------------------------
bool TriangularInferior(int a[][n],int tl){ //elementos por encima de
la diagonal principal nulos
        int i=0,triangular=1;
        while(i<tl and triangular){            int j=i+1;
            while(j<tl and a[i][j]==0) j++;
            (j<tl)? triangular=0:i++;}
        return triangular;}
//-----------------------------------------------------
bool Diagonal(int a[][n],int tl){
            int i=0,triangular=1;
    while(i<tl and triangular){    int j=0;
        while(j<tl and triangular){
```

```
        if(i==j){
            if(a[i][j]==0){triangular=0;}
        }
        else{
            if(a[i][j]!=0){triangular=0;}
        }
        j++;}
    i++;}
return triangular;}
//-----------------------------------------------------
bool marco(int a[][n],int tl){
                int i=0,marco=1;
    while(i<tl and marco){int j=0;
        while(j<tl and marco){
        if(i==0 or j==0 or j==tl-1 or i==tl-1){
                        (a[i][j]!=0)? j++: marco=0;}
        else{(a[i][j]!=0)? marco=0:j++;}}
        i++;}
return marco;}
//-----------------------------------------------------
bool filanula(int a[][n],int tl){
    int i=0,fnula=0;
    while(i<tl and !fnula){int j=0;
        while(j<tl and a[i][j]==0) j++;
    (j<tl)? i++:fnula=1;}
return fnula;}
//-----------------------------------------------------
bool columnaNula(int a[][n],int tl){
    int i=0,cnula=0;
    while(i<tl and !cnula){int j=0;
        while(j<tl and a[j][i]==0) j++;
    (j<tl)? i++:cnula=1;}
return cnula;}
//-----------------------------------------------------

void caracterizar(int a[][n],int tl){

cout<<boolalpha<<"Matriz nula?: "<<esNula(a,tl)<<endl;
cout<<boolalpha<<"Matriz simetrica?: "<<simetrica(a,tl)<<endl;
cout<<boolalpha<<"Diagonal Nula?: "<<DiagonalNula(a,tl)<<endl;
cout<<boolalpha<<"TriangularSuperior?: "<<TriangularSuperior(a,tl)
<<endl;
cout<<boolalpha<<"Triangular inferior?: "<<TriangularInferior(a,tl)
<<endl;
cout<<boolalpha<<"Diagonal?: "<<Diagonal(a,tl)<<endl;
cout<<boolalpha<<"marco?: "<<marco(a,tl)<<endl;
cout<<boolalpha<<"filaNula?: "<<filanula(a,tl)<<endl;
cout<<boolalpha<<"columnaNula?: "<<columnaNula(a,tl)<<endl;
```