```
#include <iostream>
#include <cstdlib>
#include <cmath>
using namespace std;
int getRnd(int max){
  return    rand()%max;
}
/**        Esta función recibe una matriz y rellena sus elementos con valores aleatorios en el rango (0,5). Puede usar la función:
  */
void random_fill(float m[5][5]){
        for(int i=0;i<5;i++){
                for(int j=0;j<5;j++){
                        m[i][j]=getRnd(5);
                }
        }
}

/**Imprime el vector
  */
void imprime(float m[],int n){
        for(int i=0;i<5;i++){
                cout<<m[i] <<" ";
        }
        cout<<endl;
}

/**Imprime la matriz
  */
```

```
void imprime(float m[5][5]){
	for(int i=0;i<5;i++){
		for(int j=0;j<5;j++){
			cout<<m[i][j]<<" ";
		}
		cout<<endl;
	}
}
/** Devuelve el número de veces que el valor val aparece en la matriz.
  */
int find(float m[5][5],float val)
{
int count=0;
for(int i=0;i<5;i++){
	for(int j=0;j<5;j++){
		if (m[i][j]==val){
		    count++;
		}
	}
}
return count;
}

/** Transpone la matriz pasada como argumento.
  */
void transpose(float m[5][5]){
  for(int i=0;i<5;i++){
      for(int j=i+1;j<5;j++){
          float aux=m[i][j];
```

```
            m[i][j]=m[j][i];
            m[j][i]=aux;
        }
    }
}

/**    Guarda en max el mayor elemento de cada fila.
 */
void max_row(float m[5][5],float max[5]){
    for(int i=0;i<5;i++){
            max[i]=m[i][0];
            for(int j=1;j<5;j++){
                if (m[i][j]> max[i]) max[i]=m[i][j];
            }
    }
}
/**    Guarda en min el menor elemento de cada columna.
 */
void min_col(float m[5][5],float min[5]){
    for(int i=0;i<5;i++){
            min[i]=m[0][i];
            for(int j=1;j<5;j++){
                if (m[j][i]> min[i]) min[i]=m[j][i];
            }
    }

}
/**
```

```
  *     La función imprime por pantalla la suma de los elementos pares e impares de la matriz. Debido a que los elementos son reales, será necesario utilizar solo la parte entera de los valores.

  *     Para ello, utilizar la función round(). Esta función devuelve el entero más cercano al flotante pasado como parámetro.

  */

void odd_and_even_sum(float m[5][5]){

    int nodd=0,neven=0;

    for(int i=0;i<5;i++){

            for(int j=0;j<5;j++){

                int vi=round(m[i][j]);

                if (vi%2==0) {nodd++;}

                else {neven++;}

            }

    }

    cout<<"Numero de pares ="<< nodd<<" e impares="<< neven<<endl;

}


/**La función hace la suma matricial tal que m=m+m2;

  */


void sum(float m[5][5],float m2[5][5]){

          for(int i=0;i<5;i++){

            for(int j=0;j<5;j++){

                m[i][j]=m[i][j]+m2[i][j];

            }

        }

}

/* La función hace la multiplicación matricial tal que m3=m*m2;

  */
```

```
  void multiplica(float m[5][5],float m2[5][5],float m3[5][5]){
            for(int i=0;i<5;i++){
             for(int j=0;j<5;j++){
                 m3[i][j]=0;
                 for(int k=0;k<5;k++){
                     m3[i][j]=m3[i][j]+ m[i][k]*m2[k][j];
                 }
             }
            }
  }
/**La función devuelve la traza de la matriz pasada.
  */
float trace(float m[5][5]){
  float T=0;
  for(int i=0;i<5;i++){
      T=T+m[i][i];
  }
  return T;
}

/** Calcula la media y desviación típica los elementos de la matriz y la imprime por pantalla
  */
void stats(float m[5][5] ){
    float mean=0;
    //calculate mean first
            for(int i=0;i<5;i++){
             for(int j=0;j<5;j++){
                        mean=mean+m[i][j];
             }
```

```
        }
        mean=mean/25.;
        //now, dev
        float dev=0;
        for(int i=0;i<5;i++){
         for(int j=0;j<5;j++){
                dev=dev+ ((mean- m[i][j])*(mean- m[i][j]));
         }
        }
        dev=sqrt(dev/25.);
        cout<<"mean="<<mean<<" dev="<<dev<<endl;

}

/** Normaliza la matriz. Para ello, resta a cada elemento la media y lo divide por  la desviación típica. Es decir
norm(i,j)= (m(i,j) - mean) / dev;
*/

void normalize(float m[5][5]){
   float mean=0;
   //calculate mean first
        for(int i=0;i<5;i++){
         for(int j=0;j<5;j++){
                mean=mean+m[i][j];
         }
        }
        mean=mean/25.;
        //now, dev
```

```
        float dev=0;
        for(int i=0;i<5;i++){
         for(int j=0;j<5;j++){
                dev=dev+ ((mean- m[i][j])*(mean- m[i][j]));
         }
        }
        dev=sqrt(dev/25.);

        //normalize
        for(int i=0;i<5;i++){
         for(int j=0;j<5;j++){
                m[i][j]=(m[i][j]- mean)/dev;
         }
        }
}

int main(){
  float mat[5][5];
  //ejercicio 1
  srand(time(0));//inicializa el generador de numeros aleatorios
  random_fill(mat);
  cout<<"Matriz generada"<<endl;
  imprime(mat);

  //ejercicio 2
  int number=getRnd(5);//devuelve un numero aleatorio
  int nn=find(mat,number);
  cout<<"EL numero "<<number<<" se repite "<<nn<<" veces en la matriz"<<endl;
```

```
//ejercicio 3
transpose(mat);
cout<<"Matriz Transpuesta"<<endl;
imprime(mat);

//ejercicio 5
float max_r[5];
max_row(mat,max_r);
cout<<"Los maximos por fila son       :";
imprime(max_r,5);

  //ejercicio 6
float min_c[5];
min_col(mat,min_c);
cout<<"Los minimos por columna son:";
imprime(min_c,5);

//ejercicio 7
odd_and_even_sum(mat);

//ejercicio 8
float mat2[5][5];
random_fill(mat2);
cout<<"Matriz 2 "<<endl;
imprime(mat2);
//ejercicio 8(suma)
sum(mat,mat2);
cout<<"Matriz suma"<<endl;
imprime(mat);
```

```
  //ejercicio 9 (multiplicacion)
  float mat_m[5][5];
  multiplica(mat,mat2,mat_m);
  cout<<"Multiplicacion "<<endl;
  imprime(mat_m);
  //ejercicio 5(traza)
  cout<<"La traza es "<<trace(mat_m)<<endl;
  //Ejercicio 11 (stats)
  stats(mat_m);

  //Erjercicio 12 (normalize)
  normalize(mat_m);
  cout<<"Matrix normalizada"<<endl;
  imprime(mat_m);


}
```