ejercicio 1:

```
#include <stdio.h>

#define FILAS 6
#define COLUMNAS 5

int main() {
    // Declaración e inicialización de la matriz
    int matriz[FILAS][COLUMNAS] = {0};

    // Proceso de carga de valores en la matriz
    while (1) {
        // Declaración de variables locales
        int fila, columna, valor;

        // Ingreso de la fila
        printf("Ingrese el número de fila (0 para finalizar): ");
        scanf("%d", &fila);

        // Verificar si se debe finalizar el ingreso
        if (fila == 0) {
            break;
        }

        // Validar la fila ingresada
        if (fila < 1 || fila > FILAS) {
            printf("Error: El número de fila debe estar entre 1 y %d. Ingrese nuevamente.\n", FILAS);
            continue;
        }

        // Ingreso de la columna
        printf("Ingrese el número de columna: ");
        scanf("%d", &columna);

        // Validar la columna ingresada
        if (columna < 1 || columna > COLUMNAS) {
            printf("Error: El número de columna debe estar entre 1 y %d. Ingrese nuevamente.\n",
COLUMNAS);
            continue;
        }

        // Ingreso del valor a cargar
        printf("Ingrese el valor a cargar en la posición [%d][%d]: ", fila, columna);
        scanf("%d", &valor);

        // Cargar el valor en la matriz
        matriz[fila - 1][columna - 1] = valor;
    }

    // Mostrar la matriz por filas
    printf("\nMatriz por filas:\n");
    for (int i = 0; i < FILAS; i++) {
```

```
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }

    // Mostrar la matriz por columnas
    printf("\nMatriz por columnas:\n");
    for (int j = 0; j < COLUMNAS; j++) {
        for (int i = 0; i < FILAS; i++) {
            printf("%d ", matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

ejercicio 2:

```
#include <stdio.h>

int main() {

        int matriz[3][3]={0};

        int f, c;

        for(f=0; f<3; f++){
                for(c=0; c<3; c++){
                printf("ingrese el valor de la fila %d y la columna %d:",f+1,c+1);
                scanf("%d",&matriz[f][c]);
                }
        }

        printf("las columnas pares son:\n");
        for(f=0; f<3; f++){
                printf("%d\n",matriz[f][1]);
        }

        printf("las columnas impares son:\n");
        for(f=0; f<3; f++){
                for(c=0; c<3; c++){
                        printf("%d\n",matriz[f][c]);
                        c++;
                }
        }


                return 0;
}
```

ejercicio 3

```c
#include <stdio.h>

#define FILAS 5
#define COLUMNAS 5

int main() {
    // Declaración de la matriz
    int matriz[FILAS][COLUMNAS];

    // Ingreso de la diagonal principal
    printf("Ingrese los valores de la diagonal principal:\n");
    for (int i = 0; i < FILAS; i++) {
        printf("Elemento [%d][%d]: ", i + 1, i + 1);
        scanf("%d", &matriz[i][i]);
    }

    // Ingreso del triángulo superior
    printf("\nIngrese los valores del triángulo superior:\n");
    for (int i = 0; i < FILAS - 1; i++) {
        for (int j = i + 1; j < COLUMNAS; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }

    // Ingreso del triángulo inferior
    printf("\nIngrese los valores del triángulo inferior:\n");
    for (int i = 1; i < FILAS; i++) {
        for (int j = 0; j < i; j++) {
            printf("Elemento [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }

    // Mostrar el contenido de la matriz
    printf("\nContenido de la matriz:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

ejercicio 4:

```c
#include <stdio.h>
```

```
#define FILAS 3
#define COLUMNAS 3

// Función para ordenar la matriz
void ordenarMatriz(int matriz[FILAS][COLUMNAS]) {
    // Convertir la matriz en un arreglo unidimensional
    int arreglo[FILAS * COLUMNAS];
    int k = 0;

    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            arreglo[k++] = matriz[i][j];
        }
    }

    // Ordenar el arreglo
    for (int i = 0; i < FILAS * COLUMNAS - 1; i++) {
        for (int j = 0; j < FILAS * COLUMNAS - i - 1; j++) {
            if (arreglo[j] > arreglo[j + 1]) {
                // Intercambiar elementos si están en el orden incorrecto
                int temp = arreglo[j];
                arreglo[j] = arreglo[j + 1];
                arreglo[j + 1] = temp;
            }
        }
    }

    // Actualizar la matriz con los valores ordenados
    k = 0;
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            matriz[i][j] = arreglo[k++];
        }
    }
}

// Función para mostrar la matriz
void mostrarMatriz(int matriz[FILAS][COLUMNAS]) {
    printf("\nContenido de la matriz ordenada:\n");
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("%d\t", matriz[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // Declaración de la matriz
    int matriz[FILAS][COLUMNAS];

    // Ingreso de datos
```

```
    printf("Ingrese los valores para la matriz de %d filas por %d columnas:\n", FILAS, COLUMNAS);
    for (int i = 0; i < FILAS; i++) {
        for (int j = 0; j < COLUMNAS; j++) {
            printf("Ingrese el valor para la posición [%d][%d]: ", i + 1, j + 1);
            scanf("%d", &matriz[i][j]);
        }
    }

    // Ordenar y mostrar la matriz
    ordenarMatriz(matriz);
    mostrarMatriz(matriz);

    return 0;
}
```

ejercicio 5:

```
#include <stdio.h>

#define SORTEOS 3
#define NUMEROS_POR_SORTEO 6
#define MAX_NUMERO 42

// Función para ordenar un arreglo de números de menor a mayor
void ordenarArreglo(int arreglo[], int longitud) {
    for (int i = 0; i < longitud - 1; i++) {
        for (int j = 0; j < longitud - i - 1; j++) {
            if (arreglo[j] > arreglo[j + 1]) {
                // Intercambiar elementos si están en el orden incorrecto
                int temp = arreglo[j];
                arreglo[j] = arreglo[j + 1];
                arreglo[j + 1] = temp;
            }
        }
    }
}

int main() {
    // Declaración de variables
    int sorteos[SORTEOS][NUMEROS_POR_SORTEO];
    int ocurrencias[MAX_NUMERO + 1] = {0};  // Inicializar el arreglo de ocurrencias
    int numeroMasFrecuente[MAX_NUMERO + 1] = {0};
    int n;

    // Ingreso de números para cada sorteo
    for (int i = 0; i < SORTEOS; i++) {
        printf("Ingrese los 6 números del sorteo %d (de 0 a 42 inclusive, sin repetir):\n", i + 1);

        // Validar que no se repitan números
        for (int j = 0; j < NUMEROS_POR_SORTEO; j++) {
            do {
                printf("Número %d: ", j + 1);
                scanf("%d", &sorteos[i][j]);
```

```
            // Verificar que el número esté en el rango permitido
            if (sorteos[i][j] < 0 || sorteos[i][j] > MAX_NUMERO) {
                printf("Error: El número debe estar entre 0 y %d.\n", MAX_NUMERO);
            }
            // Verificar que el número no se haya ingresado antes
            else if (ocurrencias[sorteos[i][j]] > 0) {
                printf("Error: Este número ya fue ingresado en este sorteo.\n");
            }
            // Si todo está bien, actualizar el arreglo de ocurrencias
            else {
                ocurrencias[sorteos[i][j]]++;
                break;
            }
        } while (1);
    }

    // Ordenar los números del sorteo de menor a mayor
    ordenarArreglo(sorteos[i], NUMEROS_POR_SORTEO);
}

// Mostrar los números de cada sorteo y actualizar la información del número más frecuente
printf("\nResultados de los sorteos:\n");
for (int i = 0; i < SORTEOS; i++) {
    printf("Sorteo %d: ", i + 1);
    for (int j = 0; j < NUMEROS_POR_SORTEO; j++) {
        printf("%d ", sorteos[i][j]);
        // Actualizar la información del número más frecuente
        if (ocurrencias[sorteos[i][j]] > ocurrencias[numeroMasFrecuente[0]]) {
            numeroMasFrecuente[0] = sorteos[i][j];
            numeroMasFrecuente[1] = 1;
        } else if (ocurrencias[sorteos[i][j]] == ocurrencias[numeroMasFrecuente[0]]) {
            // Si hay empate, actualizar la cantidad de veces que se repite
            numeroMasFrecuente[1]++;
        }
    }
    printf("\n");
}

// Mostrar el número más frecuente o los más frecuentes en caso de empate
if (numeroMasFrecuente[1] == 1) {
    printf("\nEl número más frecuente es: %d\n", numeroMasFrecuente[0]);
} else {
    printf("\nHubo empate entre varios números más frecuentes.\n");
}

// Ingreso de un número N
printf("\nIngrese un número N para verificar su repetición entre todos los sorteos: ");
scanf("%d", &n);

// Verificar la repetición del número N entre todos los sorteos
int repeticionesN = 0;
```

```
    for (int i = 0; i < SORTEOS; i++) {
        for (int j = 0; j < NUMEROS_POR_SORTEO; j++) {
            if (sorteos[i][j] == n) {
                repeticionesN++;
            }
        }
    }

    // Mostrar la cantidad de veces que se repite el número N
    printf("El número %d se repite %d veces entre todos los sorteos.\n", n, repeticionesN);

    return 0;
}
```

ejercicio 6:

```
#include <stdio.h>

#define MAX_PERSONAS 10

// Definición de la estructura para representar a una persona
struct Persona {
    int numeroSocio;
    int edad;
};

// Función para intercambiar dos estructuras Persona
void intercambiarPersonas(struct Persona *a, struct Persona *b) {
    struct Persona temp = *a;
    *a = *b;
    *b = temp;
}

// Función para ordenar un arreglo de estructuras Persona por edad
void ordenarPorEdad(struct Persona personas[], int longitud) {
    for (int i = 0; i < longitud - 1; i++) {
        for (int j = 0; j < longitud - i - 1; j++) {
            if (personas[j].edad > personas[j + 1].edad) {
                // Intercambiar personas si están en el orden incorrecto
                intercambiarPersonas(&personas[j], &personas[j + 1]);
            }
        }
    }
}

int main() {
    // Declaración de variables
    struct Persona personas[MAX_PERSONAS];
    int cantidadPersonas = 0;

    // Ingreso de datos
    while (cantidadPersonas < MAX_PERSONAS) {
        // Ingreso del número de socio
```

```
        printf("Ingrese el número de socio (o 0 para finalizar): ");
        scanf("%d", &personas[cantidadPersonas].numeroSocio);

        // Verificar si se debe finalizar la carga
        if (personas[cantidadPersonas].numeroSocio == 0) {
            break;
        }

        // Ingreso de la edad
        printf("Ingrese la edad: ");
        scanf("%d", &personas[cantidadPersonas].edad);

        // Incrementar la cantidad de personas ingresadas
        cantidadPersonas++;
    }

    // Ordenar el arreglo de personas por edad
    ordenarPorEdad(personas, cantidadPersonas);

    // Mostrar la lista ordenada por edad
    printf("\nLista ordenada por edad:\n");
    for (int i = 0; i < cantidadPersonas; i++) {
        printf("Número de socio: %d, Edad: %d\n", personas[i].numeroSocio, personas[i].edad);
    }

    return 0;
}

:
```