1. Diseñar un programa que lea datos del teclado, los almacene en un array y, a continuación, visualice los elementos que ocupan las posiciones pares.

2. Crea un array, rellénalo con valores desde el teclado. Muestra sus elementos indicando para cada uno si es par o impar.

3. Crear un programa que rellene por teclado un array de enteros y nos devuelva la suma de todos. El tamaño del array se le solicita al usuario.

4. Realizar un programa para sumar y mostrar los elementos impares de un array cargado desde teclado.

5. Mostrar, contar y sumar los elementos pares que ocupan las posiciones impares de un array cargado desde teclado. Mostrar las posiciones que ocupan dichos elementos en el array.

6. Crea un array, rellénalo con valores desde el teclado. Copia sus elementos al revés en otro array y muestra los elementos de los dos arrays.

7. Realiza dos métodos: el primero recibe como parámetro un array (por referencia) y el segundo un elemento del array (por valor). Los parámetros por referencia se ven afectados por el código mientras que los recibidos por valor no, porque reciben una copia, no una referencia. Demuestra que con el primer método el array queda modificado en el programa y que con el segundo método, el elemento del array no queda modificado ya que recibe una copia del elemento.

8. Dado un array de caracteres, insertar un carácter en una posición concreta, desplazando el resto de caracteres una posición. Controlar que existe espacio en el array para insertar dicho carácter.

9. Crea un array, rellénalo con valores leídos desde el teclado. Imprime el número de veces que aparece un número indicado por el usuario y la posición en el array.
   Por ejemplo:
   Si el array tiene 5 elementos con los siguientes valores: 1, 2,4,2,1
   Y el usuario quiere saber cuantas veces aparece el 2, en la pantalla hay que mostrar la respuesta en el siguiente formato:
   El numero 2 se repite 2 veces.
   Las posiciones que contienen el valor 2 son: 1,3.

10. Realiza una función recursiva que calcule si un array unidimensional de números es capicúa.

11. Rellena mediante el teclado una matriz bidimensional de enteros. Calcula y muestra en pantalla la suma de cada fila y de cada columna. El tamaño del array se le solicita al usuario. El número de filas no tiene por qué ser igual al número de columnas.

12. Dado un array de tres dimensiones de números decimales, mostrar cuál es el elemento mayor y su posición.

13. Copia un array tridimensional de enteros a otro de mayor tamaño.

14. Un array bidimensional a de N filas y N columnas es **simétrico** si sus elementos satisfacen la condición a[i,j] = a[j,i] para todo i,j, es decir, si se obtiene la misma tabla al cambiar las filas por las columnas. Escribe un programa que determine si un array de ese tipo es simétrico.
    Ejemplo:

1	2	3	4
2	5	6	7
3	6	8	9
4	7	9	10

15. Un **histograma** es una gráfica que muestra la frecuencia con que aparecen en una lista dada los distintos valores que la pudieran formar. Por ejemplo, si los valores de una lista pueden estar comprendidos entre 0 y 9, y la lista está formada por: 6, 4, 4, 1, 9, 7, 5, 6, 4, 2, 3, 9, 5, 6, 4, su histograma sería:

```
        *
        *   *
        * * *     *
  * * * * * * *   *
0 1 2 3 4 5 6 7 8 9
```

    Esto indica que 0 y 8 no aparecen ninguna vez, que 1, 2, 3 y 7 aparecen una vez, 5 y 9 dos veces, etc. Escriba un algoritmo que lea una lista de números comprendidos entre 0 y 9 (la lista acabará cuando se lea un número negativo) e imprima por pantalla un histograma como el anterior.

16. Escribe un programa que inicialice una matriz de NxN desde teclado y que diga si existe alguna fila exactamente igual a alguna columna, junto con los índices de las filas y columnas que son iguales.

17. Ordenar un array unidimensional según el método de la **burbuja**. Consiste en comparar pares de elementos adyacentes en un array y si están desordenados intercambiarlos hasta que estén todos ordenados. Este método consiste en ordenar el array moviendo el mayor hasta la última casilla comparando e intercambiando los números comenzando desde la casilla cero hasta situarlo en la última posición. Una vez situado el más grande, se procede a encontrar y situar el siguiente más grande comparando e intercambiando de nuevo los números desde el inicio del array, y así sigue sucesivamente hasta ordenar todos los elementos del array.
    Ejemplo de ejecución:

50	26	7	9	15	27	Array original
Primera pasada:						
26	**50**	7	9	15	27	Se intercambian el 50 y el 26
26	7	**50**	9	15	27	Se intercambian el 50 y el 7
26	7	9	**50**	15	27	Se intercambian el 50 y el 9
26	7	9	15	**50**	27	Se intercambian el 50 y el 15
26	7	9	15	27	**50**	Se intercambian el 50 y el 27
Segunda pasada:						
7	**26**	9	15	27	50	Se intercambian el 26 y el 7
7	9	**26**	15	27	50	Se intercambian el 26 y el 9
7	9	15	**26**	27	50	Se intercambian el 26 y el 15

18. Escribe un programa que halle si una matriz introducida es un **cuadrado mágico**. Un cuadrado mágico es una matriz cuadrada NxN donde las sumas de cada columna, de cada fila y de cada diagonal son iguales. Ej:

4	15	14	1
9	6	7	12
5	10	11	8
16	3	2	13

19. Realiza un programa que lea por teclado una cadena de caracteres y guarde en un array las palabras que componen dicha cadena (una palabra en cada posición del array). Por último, mostrar por pantalla el contenido de dicho array mostrando cada palabra en una línea distinta. No usar el método split.
    Hacer dos versiones:
    - Las palabras están separadas por un espacio
    - Las palabras pueden estar separadas por más de un espacio.

20. Escriba un programa que sea un **constructor de frases**.
    Hay que rellenar una matriz bidimensional NxN con palabras utilizando el método split de las cadenas. A continuación, el programa mostrará el siguiente menú:
    1. Impares o pares
    2. Búsqueda
    3. Salir

    Si se elige la opción 1, hay que sacar el siguiente submenú:
    1. Impares
    2. Pares
    3. Salir

    Si se elige la opción 1, hay que construir una frase con aquellas palabras que se encuentren en una celda donde la suma de su fila y su columna sea impar. Hay que mostrar la frase resultado por pantalla.
    Si se elige la opción 2, igual que la anterior pero que sea par.
    Si se elige la opción 3, se vuelve al menú principal. El submenú aparecerá repetidas veces hasta que el usuario decida salir.

Si se elige la opción 2 (Búsqueda) en el menú principal, hay que solicitar por teclado un carácter a buscar. El programa deberá construir una frase con aquellas palabras de la matriz donde aparezca dicho carácter, pero el carácter se tiene que sustituir por un asterisco en la frase resultante. Además, el programa debe mostrar cuántas palabras contienen dicho carácter en cada fila y cuántas en cada columna.
Si se elige la opción 3 (Salir) en el menú principal, se termina el programa. El menú principal aparecerá repetidas veces hasta que el usuario elija la opción 3.

En todas las opciones, en la frase resultante las palabras irán separadas por un espacio y la frase resultante debe almacenarse en una cadena. Para construir dicha frase, la matriz debe recorrerse por filas.

21. Realizar un programa en java para jugar con el ordenador a un juego de **adivinanza**. El programa tendrá que adivinar una letra o un número pensado por el usuario. Para ello, el juego consta de dos partes.
    En la primera parte, el programa tendrá que adivinar el tipo, que puede ser:
    - Un número del 0 al 9
    - Una vocal minúscula
    - Una consonante minúscula (sin contar la ñ)

    El programa tendrá dos intentos como máximo para averiguar ese tipo. Si no lo adivina, el juego termina y el ordenador habrá perdido.
    Si lo adivina, se continúa jugando con la segunda parte donde tendrá que adivinar la letra o el número concreto. Para ello tendrá los siguientes intentos:
    - El número entre 0 y 9: tres intentos
    - La vocal minúscula: dos intentos
    - La consonante minúscula: cinco intentos. En este caso, el usuario le irá ayudando diciéndole al ordenador si es mayor o menor según el orden alfabético.

    Si el programa, dentro de ese número de intentos lo adivina, habrá ganado, si no, habrá perdido. En cualquier caso, hay que informar al usuario.

22. Realizar un programa que dada una matriz cuadrada NxN, saque por consola en qué fila, en qué columna y en qué diagonal hay un número **capicúa**. Un número es capicúa si se lee igual empezando por el principio que empezando por el final.
    La dimensión de la matriz se le solicita al usuario, y también se le solicita que introduzca los datos en la matriz con números entre 0 y 9. El usuario debe saber en qué posición (fila y columna) de la matriz se está insertando el número, y si el número no es válido, se le pedirá repetidas veces hasta que inserte un número válido. La matriz se rellenará por filas y se mostrará por consola. Como el usuario no sabe de programación, para él, la numeración de las posiciones de la matriz empiezan por 1.
    Una vez se le dé el resultado al usuario, se le preguntará si desea crear otra matriz nueva para comprobar los capicúas. Podrá generar todas las matrices que quiera de diferentes dimensiones hasta que decida salir.
    Ejemplos:
    - N=1:

7

    La fila 1 es capicúa. La columna 1 es capicúa. La diagonal principal es capicúa. La diagonal secundaria es capicúa.
    - N=2:

5	8
8	1

La diagonal secundaria es capicúa.

- N=3:

2	7	3
5	1	9
6	7	2

La columna 2 es capicúa. La diagonal principal es capicúa.

23. El **blackjack** es un juego de cartas, propio de los casinos, que consiste en obtener 21 puntos mediante la suma de los valores de las cartas. Las cartas numéricas suman su valor, las figuras suman 10 y el as es un 11 o un 1 según interese.
Al principio se reparten 2 cartas a cada jugador. El jugador tiene la posibilidad de plantarse (quedarse con las cartas que tiene) o pedir carta, pero si se pasa de 21 pierde. Gana el que saque un blackjack, que es conseguir 21 con dos cartas. Si no hay ningún blackjack, gana el que tenga el número más alto sin pasarse de 21.
Hacer un programa para jugar al blackjack entre dos jugadores de tal manera que las cartas se asignen de manera aleatoria. La baraja con la que vamos a jugar tiene Picas ♠, Corazones ♥, Diamantes ♦ y Tréboles ♣. Las cartas irán del as al diez y habrá tres figuras: jota, reina y rey.
El programa primero jugará con el jugador 1 y cuando termine con él continuará con el jugador 2. El programa comenzará dándole dos cartas al jugador, indica el total que suponen las dos cartas y luego le irá preguntando si quiere más cartas o si se planta. Por cada carta que le dé al jugador le irá indicando el cómputo total tomando el as como 11 pero si se pasa el jugador entonces tomará el as como 1. Si llega a 21, blackjack o si se pasa, el programa no le dará opción a pedir más carta.
Los jugadores contarán con 100 euros cada uno para jugar. Antes de cada mano, los jugadores deciden el dinero que van a apostar en esa mano , de tal manera que el perdedor le tendrá que dar el dinero apostado al ganador y si es un blackjack, entonces es el doble de lo apostado. Si empatan, no hay movimientos de dinero. El programa les indicará la máxima apuesta que pueden hacer, que tendrá que ser la mitad del que tenga menos dinero, por si pierde con un blackjack, que pueda pagar. Una vez haya terminado la mano, el programa indicará quién ha ganado y el dinero que les queda a cada uno.
El programa les permitirá jugar tantas manos como quieran hasta que no quieran seguir jugando o hasta que uno de los dos se quede sin dinero.
En cada mano se vuelve a jugar con la baraja entera pero dentro de la misma mano habrá que controlar que no se repitan cartas. Habrá que indicar el número de mano por la que van.

24. Utilizando recursividad, realizar un programa que busque un número decimal en un array ordenado de números decimales. La búsqueda se hará de la siguiente manera: se buscará en la mitad del array, si se encuentra, se le da la información al usuario de que se ha encontrado y en qué posición del array. Si no se encuentra y el número a buscar es mayor que el elemento que se encuentra en la mitad del array, se buscará en la mitad superior del array, y si por el contrario es menor, se buscará en la mitad inferior del array. Y así se continuará sucesivamente hasta que se termine, en cuyo caso, se pueden dar dos posibilidades: que se encuentre o que no se encuentre. En cualquier caso, se informará al usuario del resultado, y si se ha encontrado, se le informará de la posición del array donde se encuentre teniendo en cuenta que para el usuario, las posiciones del array empiezan por 1.
El tamaño del array, el contenido y el número a buscar se solicitarán al usuario.
Ejemplo: elemento a buscar: 12.6

1.9	2.9	3.4	5.1	6.2	7.0	8.2	9.4	10.7	12.6	15.2
0	1	2	3	4	5	6	7	8	9	10

Se empieza a buscar en la mitad del array: posición 5.
12.6 es mayor que *7.0*, luego continuamos buscando en la mitad superior, es decir, desde la posición 6 hasta la 10. La mitad es la posición 8. *12.6* es mayor que *10.7*, luego continuamos buscando en la mitad superior, es decir, desde la posición 9 hasta la posición 10. La mitad es la posición 9. *12.6* es igual a *12.6*, luego, hemos encontrado el elemento. Al usuario le diremos lo siguiente:
"El elemento 12.6 se encuentra en la posición 10 del array"

25. Se trata de colocar **ocho reinas** sobre un tablero de ajedrez, de tal forma que ninguna amenace (pueda comerse) a otra. Una reina amenaza a otra pieza que esté en la misma columna, fila o cualquiera de las cuatro diagonales. Escribe un programa que determine todas las combinaciones posibles en las que se puedan acomodar las reinas.