Fundamentos de Programación
Relación de ejercicios 3

1. Realizar un programa que permita saber si dos vectores de enteros son iguales. Se asume que dos vectores de caracteres son iguales si el número de componentes útiles de ambos vectores es el mismo, y el valor de sus componentes coincide uno a uno.

2. Realizar un programa que modifique un vector, eliminando todas las repeticiones de un mismo valor. Ejemplo: Para el vector (1, 1, 2, 3, 4, 3, 2), el vector resultante sería (1, 2, 3, 4).

3. Implementar un programa que nos diga si una secuencia de caracteres es un palíndromo, es decir, que se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, `'a','b','b','a'` sería un palíndromo, pero `'a','c','b','a'` no lo sería. Si la secuencia tiene un número impar de componentes, la que ocupa la posición central es descartada, por lo que `'a','b','j','b','a'` sería un palíndromo. El programa debe leer una serie de caracteres hasta llegar al terminador `'#'` y diga si es un palíndromo. Se puede suponer que la secuencia no contiene espacios en blanco, ya que para poder leer un espacio en blanco no se puede emplear `cin >> caracter;`. Si se quiere también considerar los espacios en blanco, se puede utilizar la sentencia `caracter = cin.get();`. Cada vez que se ejecute `cin.get()` el compilador lee un carácter (incluido el espacio en blanco) desde la entrada de datos por defecto.

4. Suponiendo que disponemos de dos vectores ya ordenados ascendentemente, realizar un programa que muestre un nuevo vector ordenador con las componentes de los otros dos vectores. Por ejemplo, para los vectores de entrada (1,3,5,9,10) y (1,2,4,6,8), el vector resultante sería (1,1,2,3,4,5,6,8,9,10).

5. Implementar un programa que intercambie dos componentes de un vector de enteros. Por ejemplo, si el vector es (1,2,5,3), y el usuario intercambia las componentes 1 y 3, se quedaría como (5,2,1,3). El programa debe comprobar si las componentes son correctas.

6. Dados dos vectores de enteros, construid un programa que devuelva las veces que el segundo aparece en el primero. Por ejemplo, para los vectores (1,4,2,3,4,1,2,4,5) y (1,4,2), el programa devolvería el valor 2. Amplía el programa para que también devuelva la posición donde empiezan las coincidencias.

7. Implementar un programa que modifique un vector de caracteres eliminando la mayúsculas. Por ejemplo, si el vector es `{'S','o','Y',' ','y','O'}`, éste debe quedarse con `{'o',' ','y'}`. Utiliza dos bucles anidados.

8. Resolver el problema anterior con dos bucles anidados no es eficiente. Se propone ahora utilizar dos variables, `posicion_lectura` y `posicion_escritura` que nos vayan indicando, en cada momento, la componente que se está leyendo y el sitio dónde tiene que escribirse. Por ejemplo, supongamos que en un determinado momento la variable `posicion_lectura` vale 6 y `posicion_escritura`, 3. Si la componente en la posición 6 es una mayúscula, simplemente avanzaremos `posicion_lectura`. Por el contrario, si fuese una minúscula, la colocaremos en la posición 3 y avanzaremos una posición ambas variables. Implementad este algoritmo.

9. Para obtener una lista de todos los números primos menores que un determinado número n, se puede utilizar la *Criba de Eratóstenes*. Ese método consiste en hacer una lista de todos los números desde 2 hasta $n-1$. Tomamos el 2 y tachamos todos los múltiplos de 2. Luego tomamos el siguiente número que se encuentra después de 2 y que esté sin tachar, tachando de nuevo todos sus múltiplos. Repetimos este paso hasta que se acaben los números. Los

números que quedaron sin tachar son los que no tienen divisores (salvo el 1 y él mismo), o sea, los primos. Escribir un programa que obtenga los números primos menores que un determinado número n utilizando el método anterior.

10. Crear un `struct Permutacion` para representar una permutación, como función biyectiva de $\{1, \ldots, n\}$ en $\{1, \ldots, n\}$. Para almacenar la permutación usaremos como dato miembro un vector de enteros (la imagen de 1,2,...,n). Además, tendrá otro dato miembro de tipo entero para almacenar su longitud. Escribir un programa que:

    (a) solicite al usuario una permutación

    (b) compruebe si la permutación es correcta, es decir, que contiene todos los enteros sin repetir. Por ejemplo, (1,2,3,6,5,4) es una permutación correcta pero no lo es (1,2,6,5,3,3) (tiene el 3 repetido y le falta el 4).

    (c) calcule el cuadrado de la permutación (la composición consigo misma)

    Amplia el último punto para que calcule la potencia k-ésima de la permutación, para un k dado.

11. Considerar el siguiente `struct TipoConjunto` para almacenar conjuntos de números enteros de, como máximo, 1000 elementos:

```
struct TipoConjunto{
        int num_elem;
        int elementos[1000];
};
```

    El `struct` tiene como campos un vector de 1000 posiciones y un campo que indica el cardinal del conjunto, es decir, el número de elementos que actualmente forman parte del mismo. Además, el vector con los elementos se encuentra ordenado de forma ascendente, y después de cada modificación sobre la estructura (añadir un elemento, eliminarlo, etc) se debe conservar dicha ordenación. Implementar programas para:

    - determinar si un elemento se encuentra en un conjunto (es decir, en una variable de tipo `TipoConjunto`).
    - incorporar un nuevo elemento al conjunto.
    - calcular la unión de dos conjuntos
    - calcular la intersección de dos conjuntos.

    Se asume que el usuario inserta los datos iniciales del programa (por teclado o por redirección de la entrada desde un fichero).

12. Implementar un programa que calcule el número de secuencias ascendentes de un vector de enteros. Por ejemplo, el vector $\{2, 4, 1, 1, 7, 2, 1\}$ tiene 4 secuencias que son $\{2, 4\}$, $\{1, 1, 7\}$, $\{2\}$, $\{1\}$.

13. Construir un programa para comprobar si dos matrices son iguales.

14. Realizar un programa que tenga como entrada una matriz de reales y produzca como salida un vector con la suma de los elementos de cada fila. Realizar un programa que tenga como entrada una matriz de reales y produzca como salida un vector con la suma de los elementos de cada columna.

15. Leed desde teclado dos variables `util_filas` y `util_columnas` y leed los datos de una matriz de enteros de tamaño `util_filas x util_columnas`. Sobre dicha matriz, se pide lo siguiente:

(a) Calcular la traspuesta de la matriz, almacenando el resultado en otra matriz.

(b) La posición de aquel elemento que sea el mayor de entre los mínimos de cada fila. Por ejemplo, dada la matriz M (3 4),

```
9 7 4 5
2 18 2 12
7 9 1 5
```

el máximo entre 4, 2 y 1 (los mínimos de cada fila) es 4 y se encuentra en la posición (0, 2).

(c) Ver si existe un valor `MaxiMin`, es decir, que sea a la vez, máximo de su fila y mínimo de su columna.

(d) Leer los datos de otra matriz y multiplicar ambas matrices (las dimensiones de la segunda matriz han de ser compatibles con las de la primera para poder hacer la multiplicación)

16. Para ahorrar espacio en el almacenamiento de matrices cuadradas simétricas de tamaño $k \times k$ se puede usar un vector con los valores de la diagonal principal y los que están por debajo de ella. Por ejemplo, para una matriz $M = \{m_{ij}\}$ el vector correspondiente sería:

$$\{m_{11}, m_{21}, m_{22}, m_{31}, m_{32}, m_{33}, m_{41}, \ldots, m_{kk}\}$$

Declarar una matriz `double matriz[k][k]` en el `main`, asignarle valores de forma que sea cuadrada simétrica y construid el vector pedido. Haced lo mismo pero a la inversa, es decir, construir la matriz a partir del vector.

17. Se está diseñando un sistema web que recolecta datos personales de un usuario y, en un momento dado, debe sugerirle un nombre de usuario (login). Dicho login estará basado en el nombre y los apellidos; en concreto estará formado por los N primeros caracteres de cada nombre y apellido (en minúsculas, unidos y sin espacios en blanco). Por ejemplo, si el nombre es `Antonio Francisco Molina Ortega` y $N = 2$, el nombre de usuario sugerido será `anfrmoor`. Debe tener en cuenta que el número de palabras que forman el nombre y los apellidos puede ser cualquiera. Además, si N es mayor que alguna de las palabras que aparecen en el nombre, se incluirá la palabra completa. Por ejemplo, si el nombre es `Ana Campos de la Blanca` y $N = 4$, entonces la sugerencia será `anacampdelablan` (observe que se pueden utilizar varios espacios en blanco para separar palabras).

Implementar un programa `Codifica` que, a partir de una cadena de caracteres formada por el nombre y apellidos (separados por uno o más espacios en blanco) y un entero, calcule otra cadena con la sugerencia de login. Para probar el programa leemos una cadena de caracteres con la sentencia `getline(cin, cadena);` (acepta espacios en blanco en la palabra)

18. Realizar un programa que, dada una cadena de caracteres, encuentre las secuencias de más de un carácter espacio seguidas, y las reduzca a un único espacio. Por ejemplo, para la cadena `"hola___pepe__"`, la salida sería `"hola_pepe_"`, donde el símbolo _ representa al carácter espacio.

19. Construir un programa que, dada una cadena de caracteres y dos posiciones dentro de la cadena, x e y, devuelva otra cadena con los caracteres comprendidos entre las componentes x e y (inclusive).

20. Construir un programa que inserte una cadena de caracteres dentro de otra cadena, en una determinada posición. Por ejemplo, insertar la cadena `"caracola"` en la posición 2 de la cadena `hola`, resultaría la segunda cadena con el valor `hocaracolala`.