Normas para la entrega de la relación de ejercicios sobre pilas y colas

Esta entrega corresponde a la relación de ejercicios sobre pilas, colas y colas con prioridad.
- Debajo de cada ejercicio puedes encontrar la función que debes implementar. Además, debe haber un main con las distintas pruebas que has realizado para verificar el funcionamiento de la función.
- Si un ejercicio no funciona correctamente o se tiene claro que no resuelve todos los casos posibles, se debe explicar adecuadamente.
- Los ficheros no deben dar errores de compilación.

Cualquier entrega que no cumpla con estas normas no será tenida en cuenta.

Ejercicios:

1. Construye una función a la que se se le pase una pila P de tipo T y dos elementos x,y de tipo T y que modifique la pila de forma que todas las veces que aparezca x se substituya por y (quedando la pila en el mismo estado en el que estaba anteriormente). Para este ejercicio no se podrán utilizar iteradores.
   - void substituye(stack<T> &P, const T &x, const T &y)
2. Implementa un función para determinar si una expresión contenida en un string tiene una configuración de paréntesis correcta. Debe tener un orden lineal
   - bool parentesis_correctos(string expresion)
3. Implementa una función insertar(P, pos, x) que inserte un elemento en una pila P en la posición pos. La pila debe quedar como estaba antes de insertar el elemento (salvo por el nuevo elemento)
   - void insertar(stack<T> &P, int pos, const T &x)
4. Implementa una función insertar(Q, pos, x) que inserte un elemento en una cola Q en la posición pos. La cola debe quedar como estaba antes de insertar el elemento (salvo por el nuevo elemento)
   - void insertar(queue<T> &P, int pos, const T &x)
5. Usando una pila y una cola, implementa una función que compruebe si un string es un palíndromo.
   - bool palindromo(string cad)
6. Implementa un TDA cola usando como representación dos pilas.
7. Implementa el TDA pila usando dos colas. ¿Qué orden de eficiencia tienen las operaciones push y pop? (Incluye la respuesta como un comentario al final del fichero)
8. Se llama expresión en postfijo a una expresión matemática en la que cada operación aparece con sus dos operandos seguidos por el operador. Hay un espacio entre cada dos elementos. Por ejemplo:

   2 3 + 5 *

   Escribe una función que evalúe una expresión en postfijo.

   a b ^ c * d / e + donde a = 5, b = 3, c = d = 2, e = 9

   a b c d e + * + − donde a = 12, b = 4, c = 7, d = 5, e = 2

   a b + c d * + e * donde a = 2, b = 6, c = 3, d = 5, e = 9

   - T evaluar(string expresion, pair<char, T> *variables[], int num_variables)

9. Dada una matriz que representa un laberinto, construye una función que determine si se puede llegar desde la entrada hasta la salida. Esta matriz tendrá una ’E’ en la entrada, una ’S’ en la salida, un ’0’ en las casillas por las que se pueda pasar y un ’1’ en las que no. No se puede ir en diagonal.
    - bool salida_laberinto(char laberinto[][], int n_filas, int n_cols)
10. Un tipo ventana es un tipo de dato que permite insertar un elemento, mover derecha, mover izquierda, borrar elemento y que se implementa usando dos pilas. Implementa ese tipo de dato con las operaciones comentadas.
11. Implementa una cola con prioridad de un tipo struct con (apellidos, nombre, prioridad) de forma que los datos salgan de acuerdo a ese tercer campo prioridad.
12. Implementa una cola con prioridad que contenga strings y de la que salgan primero las cadenas de caracteres más largas y que en caso de igualdad salgan por orden alfabético.
13. Implementa una cola con prioridad que contenga strings y de la que salgan primero las cadenas de caracteres que tengan más vocales y que en caso de igualdad salgan por orden alfabético.