Práctica 5 – Pilas y Colas

1. Desarrollar las implementaciones de los siguientes TDAs:
   a. Pila estática
   b. Pila dinámica
   c. Cola estática (no circular)
   d. Cola dinámica
2. Ingresar en una pila los caracteres almacenados (uno por línea) en un archivo de texto. Mostrar el contenido de la pila.
   a. Utilizar la implementación estática.
   b. Utilizar la implementación dinámica.
   c. Mantener la información en la pila.
3. Ingresar los números de DNI de los votantes de una mesa a medida que van votando y mostrar los últimos N (es dato) votantes
   a. en orden inverso, comenzando por el último votante.
   b. en el orden que sufragaron.
4. Ingresar en una cola los números almacenados (uno por línea) en un archivo de texto. Mostrar el contenido de la cola.
   a. Utilizar la implementación estática no circular.
   b. Utilizar la implementación dinámica.
   c. Mantener la información en la cola.
5. Corregir los errores en el siguiente código manualmente, el cual carga una cola y luego calcula y muestra la suma de sus elementos.

```
#include <stdio.h>
#include "tdacola.h"
void leerCola(TCOLA *c);

int main(void) {
    TCOLA  cola;
    TELEMENTOC n, suma;

    iniciac(cola);
    leerCola(&cola);
    suma == 0;
    while(!vaciac(*cola)) {
         sacac(cola,  &n);
         suma -= n;
    }
    printf("%d", suma);
    return 0;
}

void leerCola(TCOLA *c) {
    TELEMENTOC n;
    iniciac(&c);
    while(scanf("%d", n))
        ponec(c, &n);
}
```

6. Desarrollar subprogramas (con estructuras iterativas) que trabajando sobre una pila de enteros:
   a. cuente la cantidad de ceros que contiene.
   b. obtenga el valor promedio (sin perder la información de la pila)

   c. calcule los valores máximo y mínimo de la pila (sin perder la información de la pila)
   d. quite todos los valores mayores al último.

7. Rehacer los ejercicios 2 y de 3 de pilas mediante subprogramas recursivos. Las pilas no deben perder su información.

8. Transformar, mediante un subprograma recursivo, una pila de manera que tenga los mismos datos pero cambiados de signo.

9. Desarrollar un programa que sume dos números enteros positivos, permitiendo una cantidad de dígitos mayor a la que soportan los tipos numéricos. Los números pueden tener diferente magnitud y deben ser ingresados comenzando desde su dígito más significativo (en orden). Utilizar pilas de caracteres.

10. Ingresar una secuencia de caracteres terminada en punto que representa una expresión aritmética.
   a. Comprobar que los paréntesis estén balanceados, de no ser así informar si falta izquierdo o derecho. Los paréntesis son los únicos símbolos a controlar. Ejemplo correcto: (( ) ( )). Ej. incorrectos: (( ) ( ) ; ( )) (; ( )) (
   b. Agregar a la expresión la presencia y comprobación de corchetes y llaves. Ejemplos correctos: { ( ) [( )] } ; ({ }). Ej. incorrectos: ({) }; [( ) ; ( )]

11. Simular una fila de empleados para usar un horno microondas en el comedor de una empresa. Para cada empleado se tiene: su número de legajo, el momento de arribo (en segundos a partir de las 12:00) y la cantidad de segundos que piensa usar el horno. El programa debe mostrar para cada empleado el tiempo en que empieza y termina el uso del horno. Al finalizar mostrar tiempo promedio de espera y legajo con mayor tiempo de espera.

12. Desarrollar un programa que reciba una expresión aritmética que contemple las cuatro operaciones básicos (+, -, * y /) en notación postfija (polaca invertida) y devuelva su valor, a menos que ocurra un error porque la expresión no sea válida. Las expresiones postfijas no necesitan paréntesis para especificar el orden de las operaciones. Ej: 5 *(1+3) en notación posfija es 5 1 3 + *. Los operandos se limitan a números de un solo dígito y son leídos como caracteres, de la misma manera que los operadores.

13. Desarrollar la implementación del TDA cola estática circular.

14. Utilizar las estructuras que considere adecuadas para determinar si una palabra almacenada en una cola es palíndroma. Se conoce de antemano la cantidad de letras de la palabra (N).

15. Un sistema de inscripción a una maratón toma los datos de una cola de competidores (apellido y nombre) y genera las identificaciones apilándolas según la inicial de los apellidos; debido a la gran cantidad de inscriptos se pide implementar una pila por cada inicial. Mostrar los datos de la pila de los apellidos que comienzan con una letra solicitada al operador.

16. Simular un juego de naipes donde N jugadores van tomando un naipe del mazo por turno. El juego finaliza cuando el mazo queda vacío. El puntaje de cada jugador consiste en la sumatoria de los números de los naipes que fue tomando. Si el palo del naipe tomado coincide con el palo del levantado en el turno anterior el número se duplica. Implementar utilizando las estructuras (pilas y/o colas) que considere necesarias.