Guía de Ejercicios #10 - Estructuras avanzadas

```
Advertencia

La resolución conjunta o grupal de los ejercicios aquí presentes no está
permitida, excepto en la medida en que puedas pedir ayuda a tus compañeros
de clase y a otras personas, y siempre que esa ayuda no se reduzca a que
otro haga el trabajo por vos.

El código fuente entregado por un estudiante debe ser escrito en su
totalidad por dicha persona.
```

Ejercicio 10.1

Utilizando la siguiente estructura:

```
typedef struct Nodo {
char usuario [ 30 ] ;
int clave ;
} Nodo ;
```

Se desea implementar un módulo de manejo de pilas. Para ello, dicho módulo debe poseer las siguientes funciones:

int push(Nodo** pila, int* largo, Nodo nodo)	Inserta un nodo en la pila.
int pop(Nodo** pila, int* largo, Nodo* nodo)	Quita un nodo de la pila.
int espiar(Nodo* pila, int largo, Nodo* nodo)	Devuelve el valor del próx. nodo a quitar de la pila.
int mostrar(Nodo* pila, int largo)	Imprime el contenido de la pila.

Ejercicio 10.2

Utilizando la siguiente estructura:

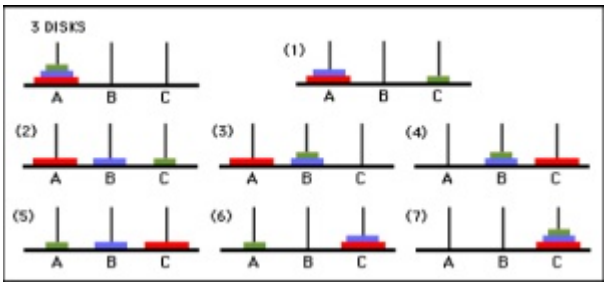
```
typedef struct Nodo {
char usuario [ 30 ] ;
int clave ;
} Nodo ;
```

Se desea implementar un módulo de manejo de colas. Para ello, dicho módulo debe poseer las siguientes funciones:

int encolar(Nodo** cola, int* largo, Nodo nodo)	Inserta un nodo en la cola.
int desencolar(Nodo** cola, int* largo, Nodo* nodo)	Quita un nodo de la cola.
int espiar(Nodo* cola, int largo, Nodo* nodo)	Devuelve el valor del nodo a quitar de la cola.
int mostrar(Nodo* cola, int largo)	Imprime el contenido de la cola.

Ejercicio 10.3

Impremente un código que resuelva el juego de las Torres de Hanoi para 3 discos utilizando **pilas**. Para esto, es necesario tener tres pilas: una origen, en la cual se encuentran los 3 discos al inicio; una auxiliar y; una pila de destino que, es en la que deberán encontrarse los discos al finalizar el programa.



Reglas del juego:

- Sólo se puede mover un disco a la vez, cada disco se moverá cuando se oprima la tecla 's'.
- Un disco de mayor tamaño no puede quedar arriba de uno más chico
- Solo se puede desplazar el disco que se encuentre arriba de los otros en una pila

Ejercicio 10.4

Utilizando la siguiente estructura:

```
typedef struct Nodo {
char usuario [ 30 ] ;
int clave ;
} Nodo ;
```

Se desea implementar un módulo de manejo de arreglos dinámicos (vectores). Para ello, dicho módulo debe poseer las siguientes funciones:

- int insertar(Nodo** vector, int* largo, Nodo nodo, int posicion);

> Inserta un nodo en la posición indicada.

- int eliminar(Nodo** vector, int* largo, Nodo* nodo, int posicion);

> Elimina el nodo presente en la posición indicada.

- int obtener(Nodo* vector, int largo, Nodo* nodo, int posicion);

> Devuelve el valor del nodo presente en la posición indicada.

- int buscar(Nodo* vector, int largo, Nodo nodo, int* posicion);

> Devuelve la posición del nodo con un valor coincidente al indicado.

- int mostrar(Nodo* vector, int largo);

> Imprime el contenido del vector.

Ejercicio 10.5

Implemente un programa que permita manejar la lista de espera de un servicio telefonico:
Al ingresar por consola la letra C, se podrá ingresar un nuevo cliente que poseerá: un nombre, apellido, edad y, el sistema le entregará un *número de atención*.
Al ingresar por consola la letra A, el sistema permite 'atender' a los clientes, con lo cual al ingresar algún numero de atención el sistema muestra los datos de dicho cliente, si existe, y da la opcion de retirarlo de la lista.

Ejercicio 10.6

Para este ejercicio deberá desarrollar un programa que simule un colectivo, donde entran y salen pasajeros del mismo, como si fuera una cola. Para dicho programa usted tendrá un menú en el cual tendrá dos opciones para el chofer: abrir la puerta delantera y, abrir la puerta trasera.
Al abrir la puerta delantera los pasajeros ingresan a la cola mientras que al abrir la puerta trasera los pasajeros salen. En este caso el tiempo de ingreso y egreso será de un segundo, por ejemplo: luego de que ingrese un pasajero, pasará **un segundo** antes de que ingrese otro a no ser que se salga del menu de ingreso(se cierre la primer puerta).
Considere a cada pasajero como una estructura de datos que contiene un nombre y un número de ticket. El nombre deberá generarse aleatoriamente mientras que el ticket es un contador que se reinicia con el inicio del programa.

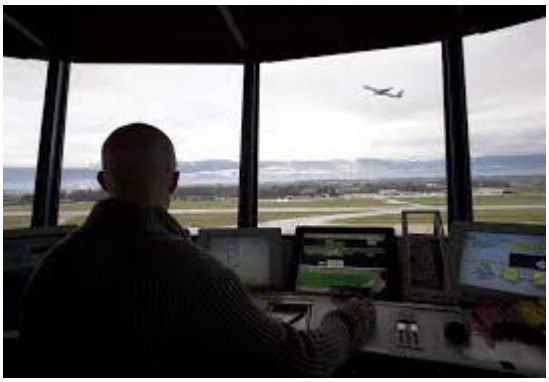
Nota: Solo es posible abrir una puerta por vez y el colectivo tiene un límite de 30 pasajeros.

Ejercicio 10.7

Escribir una programa que simule el trabajo de una torre de control de un aeropuerto con una sola pista de aterrizaje. El avión debe considerarse como una estructura que contenga el nombre del avión(por ejemplo "AR156") y su condicion de espera.
Deberá generar 4 funciones:

- Para avisar que un avión quiere aterrizar en la pista: nuevo_arribo("AR156");//recibe el nombre del avion
- Para avisar que un avión quiere despegar de la pista: nueva_partida("AR156");//recibe el nombre del avion
- Para mostrar el estado actual de espera (los que esperan para aterrizar y los que esperan para despegar): ver_estado();
- Para asignar la pista a un avión: asignar_pista();//quita un avión de la cola

Nota: Los aviones que están esperando para aterrizar tienen prioridad sobre los que están esperando para despegar.



Referencias

Algunos ejercicios fueron obtenidos y adaptados de:

- Guía de Trabajos Prácticos 2011 - Informática I - Departamento de Electrónica - UTN FRBA
- Guía de Trabajos Prácticos 2019 - Informática I - Departamento de Electrónica - UTN FRBA
- Guía de Ejercicios - Algoritmos y Programación I - UBA FIUBA