Relación de Problemas Tema 2

1.- Implementar un procedimiento con un parámetro de tipo puntero a entero que muestre el valor de la variable a la que apunta el puntero **(Acceso a variable anónima)**

2.- Implementar un procedimiento `intercambiar` con dos parámetros de tipo puntero a entero. Intercambiar su valor. Mostrar los valores antes y después de llamar al procedimiento **(Uso de parámetros de E/S)**

3.- Implementar un procedimiento `leerEnteros` que tenga dos parámetros, un parámetro que sea un puntero a una zona de memoria de números enteros y otro parámetro que sea la cantidad de valores a leer. El procedimiento introducirá en la zona de memoria apuntada por el puntero los valores leídos por teclado. Definir en el programa principal un array e invocar a este procedimiento **(Uso de aritmética de punteros/uso arrays)**

4.- Realizar lo mismo del ejercicio 3 definiendo en el programa principal la zona de memoria de forma dinámica **(Uso de aritmética de punteros/gestión dinámica de memoria)**

5. Implementar un procedimiento `mostrarEnteros` con un parámetro de tipo puntero a una zona de memoria de números enteros y mostrar su contenido. En el programa principal definir el array con el que se invocará el procedimiento **(Uso de aritmética de punteros/uso arrays)**

6.- Realizar lo mismo del ejercicio 5 definiendo en el programa principal la zona de memoria de forma dinámica **(Uso de aritmética de punteros/gestión dinámica de memoria)**

7.- Definir un registro `Info` compuesto por los campos nombre del estudiante, apellidos del estudiante, número de notas disponibles para el estudiante y una zona de memoria para guardar tantos valores de tipo real como número de nota haya disponibles para el estudiante. Definir procedimientos para leer y mostrar los campos del registro. En el programa principal definir una variable que sea un puntero a un registro de tipo `Info` y reservar la memoria para guardar la información de una persona de forma dinámica. La memoria para guardar las notas del estudiante también se reservará de forma dinámica. Definir también un procedimiento para liberar la memoria asignada previamente **(Registro y gestión dinámica de memoria)**

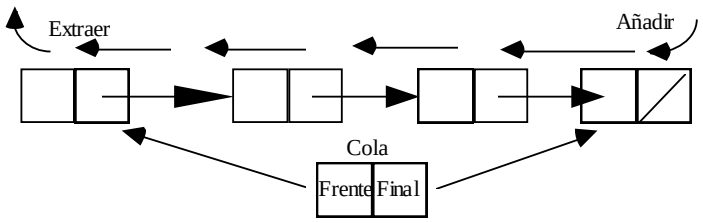
8.- Implementar la siguiente biblioteca **(Primer ejemplo de manejo de listas dinámicas y manejo de ficheros)**:

```
#ifndef __LISTA_H__
#define __LISTA_H__

// Definición de la lista enlazada
typedef struct T_Nodo* T_Lista;
struct T_Nodo {
       unsigned num;
       T_Lista sig;
};


// Crea la estructura utilizada
```

```
void Crear(T_Lista* lista);

// Destruye la estructura utilizada.
void Destruir(T_Lista* lista);

// Rellenar lista
void Rellenar (T_Lista *lista);

// Mostrar lista
void Mostrar (T_Lista lista);

// Escribir contenido de la lista en un fichero
void EscribirF(char *nombre,T_Lista lista);

// Leer datos de un fichero y crear la lista
void LeerDeFichero(char*nombre,T_Lista *l);


#endif /* __LISTA_H__ */
```

9.- Definir una biblioteca para gestión de una lista dinámica de enteros con las operaciones habituales: Crear, Mostrar, Insertar por el principio, Insertar por el final, Insertar ordenado, Eliminar un elemento, Destruir **(Segundo ejemplo de manejo de listas dinámicas)**

10. Definir una biblioteca para manejar una cola de procesos esperando para ser ejecutados. Los procesos estarán identificados por su ID (cadena de caracteres de como máximo 5 caracteres). Las operaciones que se podrán realizar serán: crear una cola de procesos, añadir un proceso a la cola, extraer un proceso de la cola, vaciar toda la cola, comprobar si la cola está vacía, mover un número determinado de procesos de una cola origen a una cola destino (la cola destino estará creada previamente y podría tener ya procesos en espera. Si hay menos procesos que los solicitados se mueven los que sea posible), mostrar el contenido de la cola por pantalla y devolver el número de procesos esperando. Se escribirá un programa principal para probar todas las operaciones anteriores. Implementar la cola como una lista enlazada apuntada por dos punteros, uno al primer nodo de la lista y otro al último nodo.



11. Modificar la implementación de la cola anterior de forma que se declare un sólo puntero cola que apunte al frente de la cola. La lista enlazada que implementa la cola es circular tal y como se muestra en la figura. El programa principal debe ser el mismo que el del ejercicio 10.