Trabajo Práctico “Estructuras y Uniones”

NOTA: Todos los ejercicios deben estar documentados siguiendo las reglas de trabajo de Doxygen. Deben tener un main con el cual se puedan probar las funciones solicitadas, como así también el Makefile correspondiente.

5.1

Escriba una función que se encargue de la carga de datos sobre una estructura “alumno” que debe contener : Nombre, Apellido, Edad, DNI y Promedio.

5.2

Escriba una función que imprima todos los datos contenidos en la estructura “alumno” previamente definida en el ejercicio 5.1

5.3

Escriba una función que remueva los duplicados de una lista simplemente enlazada de nodos de la estructura “alumno” previamente definida en el ejercicio 5.1.

5.4

Escriba una función que remueva un nodo de una lista simplemente enlazada, teniendo acceso únicamente a ese nodo.
EJEMPLO
Entrada: el nodo ‘c’ the la lista simplemente enlazada a->b->c->d->e
Resultado: No devuelve nada, pero la nueva lista simplemente enlazada queda como a->b->d->e

5.5

Escriba un programa que imprima, byte a byte, los bytes que constituyen una variable long. Utilice Uniones para su implementación.

5.6

Dada una lista circular simplemente enlazada, implementar un algoritmo que devuelva el nodo al inicio del lazo.
DEFINICIÓN Lista circular simplemente enlazada: Es una lista (corrupta) simplemente enlazada en la cual uno de los punteros a next de sus nodo, apunta a un nodo previo, de modo de crear un lazo en la lista.
EJEMPLO:

Entrada: A -> B -> C -> D -> E -> C
Salida: C

5.7

Definir un Tipo de Dato “Complex” que permita operar con números complejos. Implementar las funciones suma, resta, producto, cociente, y conjugado.

5.8

Escribir un programa que cargue un vector de estructuras de tipo:

```
struct datos {
        long legajo;
        char apellido[31];
        char nombre[31];
};
```

El ingreso de datos se hará en base a una función con el siguiente prototipo:

void carga(struct datos *);

Los datos se ingresan en un vector utilizando la función carga.
La condición de fin es legajo = 0. Escriba para esto otra función, que reciba la estructura y devuelva 1 (uno) si se cumple la condición de fin, o un 0 (cero) en caso contrario. El prototipo es:

int fin(struct dato)

Una vez generado el vector, se deberá ordenarlo en forma creciente por apellido. Para ello, debe también utilizar una función para la cual Ud. debe proponer el prototipo. Finalmente, el vector ordenado debe ser guardado, registro por registro, en el archivo binario legajos.dbf, en el directorio raíz de una disquetera. Al utilizar las funciones propuestas, Ud. notará que, al ingresar legajo=0 para salir del programa, la función carga() lo fuerza a ingresar un apellido y nombre que luego serán descartados. Sugiera una forma de solucionar esto, sin modificar el prototipo de la función carga().