Guía de Trabajos Prácticos

Esta guía contiene los Trabajos Prácticos (TP) obligatorios del curso R1042, incluyendo una guía para su entrega con el formato de presentación y un cronograma con las fechas de entrega.

Durante el transcurso del año de deberán entregarán Trabajos Prácticos, que se caracterizan por ser incrementales. Esto significa que en cada TP se irán introduciendo nuevos conocimientos con respecto al TP anterior. A su vez, se deberán utilizar las herramientas SVN, Makefile y Doxygen cuando se lo indique.

Formato de presentación

- Los archivos fuentes deben tener en todos los casos los comentarios necesarios para clarificar su lectura.
- Cada subrutina/función, debe contar con un encabezado describiendo la operación que realiza, los parámetros que espera como entrada, y los resultados que debe presentar, indicando formato y método de entrega.
- Como encabezado del programa, debe haber un comentario que explique claramente que hace dicho programa, y las instrucciones detalladas (comandos) para su compilación y "linkeo".
- TODOS los ejercicios son obligatorios.
- **La entrega de TODOS los trabajos prácticos es obligatoria para regularizar la materia.**

Trabajo Práctico N°8

- Listas y Estructuras -

Ejercicio 1

Para la siguiente estructura de datos:

```
struct sData
{
        char Nombre[30];
        char Apellido[30];
        unsigned char Edad;
}
```

Crear una estructura struct sNode que permita agrupar muchos elementos del tipo struct sData en una lista simplemente enlazada.

Ejercicio 2

Repetir el ejercicio anterior creando una estructura struct dNode para una lista doblemente enlazada.

Ejercicio 3

Repetir el ejercicio anterior para una estructura struct pNode de lista del tipo pila. Si encuentra similitudes con algo hecho anteriormente, recuerde que puede crear nuevos tipos de variable con typedef, sin tener que definir nuevamente lo mismo.

Ejercicio 4

Utilizando la estructura sNode del ejercicio 2, implemente las siguientes funciones:

```
void s_insert( struct sNode * List, void * Data);

void s_delete( struct sNode * List, struct sNode * Node);
```

La función s_insert agrega un elemento de Datos a la Lista indicada.

La función s_delete elimina el Nodo indicado de la Lista. No olvide liberar la memoria reservada para ese nodo.

Ejercicio 5

Utilizando la estructura dNode del ejercicio 3, implemente las siguientes funciones:

```
void d_insert ( struct dNode * List, void * Data);

void d_delete( struct dNode * Node );
```

La función d_insert agrega un elemento de Datos a la Lista indicada. Retorna el puntero al elemento resultante.

La función d_delete elimina el Nodo indicado.

No olvide liberar toda la memoria reservada para ese nodo.

Ejercicio 6

Utilizando la estructura pNode del ejercicio 4, implemente las siguientes funciones:

```
void push( struct pNode ** Stack, void * Data );

void * pop( struct pNode ** Stack );
```

La función push agrega un elemento de datos al tope de la pila, y modifica el puntero al stack para que apunte a ese último elemento (que es el nuevo comienzo de la lista).

La función pop retorna un puntero a los datos, y modifica el puntero al stack para redefinir el comienzo de la pila (que ahora tiene un elemento menos).

No olvide liberar la memoria correspondiente en este caso.

Ejercicio 7

Implementar una función que realice el intercambio (swap) de dos nodos de una lista simplemente enlazada.

```
void s_swap( struct sNode * List, struct sNode * node1, struct sNode * node2);
```