Autor: Eduardo Murillo, M.Sc.

Programación de Sistemas
Proyecto 1

1. Objetivo

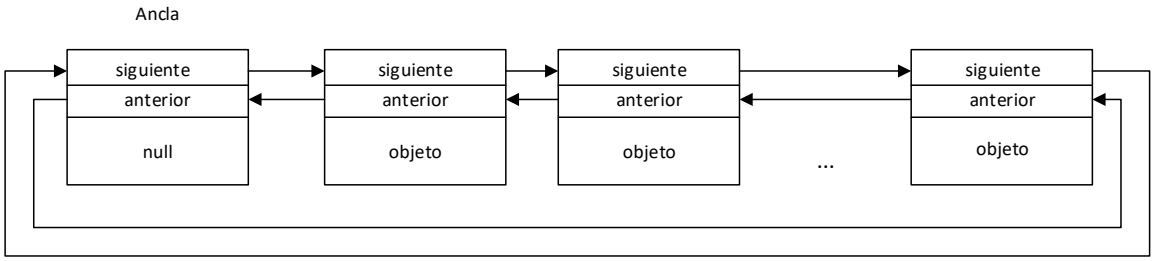
El objetivo de este proyecto es practicar la sintaxis y el uso del lenguaje C (incluido punteros), así como practicar el uso de las herramientas `gcc`, `make` y `git`.

2. Requisitos

El proyecto debe realizarse de manera **grupal**. Para este proyecto será necesario usar **Ubuntu 16.04 LTS (versión Workstation)**.

3. Descripción

En este proyecto implementaremos una lista en enlazada especial como una **librería dinámica (.so)**. La lista enlazada se **comportará** como una lista simple, pero internamente será **implementada** como una lista doblemente enlazada. Veamos un diagrama para tener una mejor idea:



En la implementación de la lista, cada elemento tiene un puntero que apuntará a al elemento que le sigue, y otro puntero que apuntará al elemento anterior. Cada elemento de la lista tiene un puntero tipo `void *` de tal forma que podemos almacenar cualquier tipo de dato en la lista.

La lista tiene un "ancla". **Esta sirve para demarcar tanto el principio y el fin de la lista**. El ancla también contiene el número de elementos que contiene la lista. Nótese que el ancla es otro elemento más de la lista, pero **NO** cuenta como elemento de la lista (simplemente es una manera conveniente de delimitar la lista).

Ya que a nivel lógico lo que queremos es una **lista simple**, hay que tener mucho cuidado al implementar determinadas funciones como `Lista_Siguiente` y `Lista_Anterior` (además de otras). Se definen dos estructuras en el código:

- `ElementoLista`: definición de cada elemento de la lista.
- `ListaEnlazada`: la lista propiamente (de las cuales se habló anteriormente).

Con respecto a `Lista_Sacar` y `Lista_SacarTodos`, estas funciones desvinculan el elemento de la lista (lo que implica cambiar los punteros del elemento anterior y siguiente al elemento que se está sacando). Las funciones `Lista_Primero`, `Lista_Ultimo`, `Lista_Anterior` y `Lista_Siguiente`, retornan un elemento de la lista (si existe), pero **NO** lo sacan de la lista.

Se proveen los siguientes archivos:

1. miLista.h → archivo cabecera que contiene las definiciones de las funciones y estructuras. **NO SE PERMITE MODIFICAR ESTE ARCHIVO.**
2. pruebaLista.c → Programa de prueba de la lista. Se lo proporciona para que puedan probar su implementación de la lista. **NO SE PERMITE MODIFICAR ESTE ARCHIVO.**

Se requerirá lo siguiente:

1. Se deberá tener las carpetas src, lib, include, obj, con los archivos correspondientes en cada carpeta.
2. Se deberá tener un **makefile** dentro de la raíz del proyecto
3. Se deberá trabajar en un repositorio de Git. **Se revisarán los commits de ambos integrantes del grupo.**
4. Se deberá crear la librería **libmilista.so**.
5. Implementar cada función declarada en el .h **en un archivo .c separado.**
6. Se deberá
    a. Compilar cada .c por separado
    b. Generar la librería .so
    c. Crear el ejecutable **prueba** enlazando pruebaLista.o con la librería .so.

El profesor debe ser capaz de escribir el siguiente comando:

```
$ make
```

Y todos los archivos deben ser generados. Luego, profesor ejecutará su programa. Noten que el nombre del ejecutable a generar debe ser **prueba**.

```
$./prueba 5000
```

Finalmente, el Makefile debe proveer la capacidad de borrar todos los archivos **.o, .so y ejecutables** generados (make clean)

4. Calificación

Las métricas de calificación están mostradas abajo. Primero, puntos serán sumados si se cumplen los siguientes requisitos.

a. Existe un Makefile +10
b. La implementación pasa las pruebas
    a. BarajarLista +15
    b. BuscarTodosEnLista +15

c.	CopiarListaHaciaAdelante	+15
d.	CopiarListaHaciaAtrás	+15
e.	CopiarListaPares	+15
f.	OrdenarLista	+15
TOTAL		**100**

Si su programa no compila por ERRORES en su código o no tiene un MAKEFILE, no ganarán puntos.

Luego, se verificará lo siguiente (puntos restados)

a.	Existe un *segmentation fault* atribuible a la lista	-10
b.	Existen *warnings* al compilar el proyecto	-5/warning
c.	Los archivos no son compilados separadamente	-20
d.	El archivo cabecera (.h) fue modificado	-20
e.	El archivo pruebaLista.c fue modificado	-20
f.	El proyecto no fue organizado por carpetas	-10
g.	No existe manera de borrar los archivos generados	-5

Se dará el siguiente crédito extra:

a.	Código comentado apropiadamente	+10

5. Entrega

El proyecto deberá ser entregado hasta el **25 de Junio de 2015, 23:59**. Se deberá proveer el repositorio de Git del proyecto con todos los archivos necesarios.