CC3301: Programación de Software de Sistemas, Primavera 2010
Profesor: Tomás Barros **Auxiliar:** Rodrigo Dueñas F.

Reglas para las Tareas

Estilo de programación en C

1. **Nombres de variables, funciones y tipos:**

   - Tienen que ser autoexplicativo.
   - Tienen que dar una idea de lo que se trata.
   - Tienen que haber una consistencia entre los nombres. No deben mezclar idiomas ni convenciones: `VariableUno, variable_uno, variableOne`.
   - No se ponen números de manera repetida dentro del código como constantes, para esto se utilizan las macros (con #define). Así no hay información redundante y pueden cambiar los parámetros del programa rápidamente.

2. **Variables:**

   - No se debe poner números de manera repetida dentro del código como constantes, para esto se utilizan las macros (con `#define`). Así no hay información redundante y pueden cambiar los parámetros del programa rápidamente.

3. **Funciones:**

   - Las funciones deben ser pequeñas en extensión. En general, si una función es de más de una pantalla es porque está haciendo demasiadas cosas y deben dividirla en sus componentes, para que sea más entendible.
   - Usen tabulación para indicar el flujo de control en sus funciones.

4. **Comentarios:**

   - Cada función necesita un comentario que señale su intención: exponiendo lo que debería hacer la función, mencionar cada argumento y el valor retornado por ésta.
   - Cada definición de `UNION` y `STRUCT` necesita un comentario que señale qué representa este nuevo tipo, qué significan sus componentes y aclaraciones sobre valores especiales o específicos sobre éstas (sólo si es necesario).
   - En el caso de que una función contenga una parte algorítmicamente compleja, debe adjuntarse un comentario explicando qué es lo que hace.

5. `.c` **versus** `.h`

   - Los archivos `.h` son para declarar, no para definir.
   - Todos los `#define` deben ir en un .h
   - Todos los tipos (`UNION` y `STRUCT`) deben ir en un `.h`
   - Todas las declaraciones de funciones deben ir en un `.h`
   - La implementación de las funciones debe ir en un `.c`

Estándares a cumplir: esto corre para todas las tareas a menos que se indique lo contrario.

1. Su código debe correr con las flags `ansi`, `pedantic` y `Wall`. Para que corra con `ansi`, su código debe cumplir ciertas características. Algunas de ellas son:
2. Su código debe correr sin leaks de memoria (esto se verifica con *Valgrind*).
3. Deben mandar todos los archivos (incluidos los del profesor), no sólo los que ustedes hicieron.
4. Deben incluir un archivo de `README.TXT` explicando qué hicieron y qué no, además deben incluir instrucciones de como compilar su tarea.
5. Deben incluir un `Makefile` que cumpla con lo siguiente:
   - Debe tener una regla `make tarea#` con # = 1, 2...etc. que compile la tarea y cree un ejecutable `tarea#`
   - Deben tener una regla `make clean` que sólo deje los archivos de texto, los archivos `.c` y los `.h` en la carpeta
6. Si se hace `make tarea#` y luego `./tarea#` debe funcionar. Si no funciona se les va a bajar puntaje.
7. Deben entregar la tarea en un archivo comprimido `tarea#.tar.gz`