INF-253 Lenguajes de Programación
Tarea 2: C

Profesor: Roberto Nicolas Diaz Urra
Ayudante Cátedras: Sebastián Godínez
Ayudante Tareas: Monserrat Figueroa - Sebastián Campos

Lunes 9 de Septiembre de 2019

1. Historia

Luego de una ardua batalla contra el señor SQL y muchas preguntas en Doomle, has logrado conseguir tu primera piedra LP del infinito. Decidido,volviste a tu casa para tomar un merecido descanso, pero grande fue tu sorpresa al encontrarla destruida y ultrajada. Tus méritos ya se estaban comenzado a esparcir por todo el mundo, por lo que ya imaginas que algunos entes estarían dispuestos a hacer lo que sea para perjudicar tu misión. Entre los escombros, encuentras una nota: "* "; lleno de determinación y con un largo camino adelante, encaminas tu viaje en busca de venganza y tu próxima piedra. Por un erudito que encontraste en tu viaje (A.K.A Inez God) sabes que la próxima piedra es la piedra del vacío, y se encuentra en Malloc City. Con punteros en tus ojos, comienzas tu nuevo viaje.

2. Objetivos

El alumno implementará el TDA Lista de manera genérica, el cual pueda almacenar distintos tipos de valor, además de implementar funciones que operen con esta estructura. Esto se realizará mediante el uso de **estructuras**, **punteros a funciones** y **punteros a void**, entre otros elementos del lenguaje.

3. Requerimientos

- El programa debe realizarse en lenguaje C.
- El programa debe compilar en los computadores del LDS.
- Se debe hacer uso de punteros a void y a funciones.
- Las funciones y estructuras propuestas deben declararse en un archivo .h e implementadas en un archivo .c.
- Para la compilación se requerirá un archivo Makefile, debe utilizarse gcc y el argumento -Wall.

4. TDA Lista

En primera instancia se debe implementar el TDA lista, para esto se utilizaran las siguientes estructuras y declaraciones de funciones.

```
struct lista {
    struct nodo* actual;
    struct nodo* head;
    struct nodo* tail;
    int length;
}
struct nodo {
    struct dato info;
    struct nodo* next;
}
struct dato {
    void* contenido;
    char tipo;
}
```

Donde `contenido` puede ser un entero, flotante u otra lista, los tipos se denotarán en el campo `tipo` como `i`, `f` o `l` respectivamente .

- `void init(struct lista* a)`: Inicializa la lista.
- `void clear(struct lista* a)`: Elimina los elementos de una lista, dejándola vacía.
- `void insert(struct lista* a, int i, dato d)`: Inserta un elemento en la posición i de la lista.
- `void append(struct lista* a, dato d)`: Agrega un elemento al final de la lista.
- `void remove(struct lista* a, int i)`: Elimina y libera la memoria de un elemento en la posición i de la lista.
- `int length(struct lista a)`: Retorna la cantidad de elementos de la lista.
- `dato* at(struct lista* a, int i)`: Retorna el elemento en la posición i de la lista.

5. Funciones a implementar

Las siguientes funciones deben operar con el TDA lista implementado anteriormente, por lo tanto deben ser capaz de soportar la anidación de listas y usos de flotantes y enteros.

- `lista* map(struct lista* a, dato (*f)(dato))` : Se aplica la operación `f` sobre cada elemento en la lista, incluyendo los elementos anidados, retornando un puntero a una lista nueva.
- `float sum(struct lista* a)`: Suma total de los elementos de la lista. En caso de una lista vacia, esta se considera como 0.
- `void print(struct lista* a)`: Se muestra en un formato adecuado la lista por pantalla, se debe lograr notar claramente las listas anidadas.
- `float average(struct lista* a)`:Se calcula el promedio total de la lista, si existe una lista anidada, se calcula primero el promedio de la lista interna y el resultado es usado para el calculo del promedio total. En caso de una lista vacía, no se considera para el calculo del promedio.

6. Archivo a entregar

- **La entrega debe realizarse en un tar.gz y debe llevar el nombre: Tarea1LP_RolIntegrante-1_RolIntegrante-2.tar.gz**
- El tar.gz debe incluir **seis** archivos:
  - lista.h : El cual contiene las declaraciones del TDA.
  - lista.c : Contiene las implementaciones del TDA.
  - funciones.h: Contiene las declaraciones de las funciones a implementar.
  - funciones.c: Contiene las implementaciones de las funciones.
  - Makefile: Archivo con los comandos de compilación del TDA y las funciones.
  - README.txt
- **El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones detalladas para la correcta compilación y utilización de su programa.**

7. Sobre la Entrega

- Se debe trabajar en grupos de dos personas.
- **El no cumplir con las reglas de entrega conlleva un descuento máximo de 30 puntos en su tarea.**
- La entrega será vía moodle y el plazo máximo de entrega es hasta el **Domingo 29 de Septiembre a las 23:55 hora moodle**.
- Por cada día de atraso se descontarán 20 puntos (10 puntos dentro de la primera hora).
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- Se deben comentar todas las funciones creadas que no estén indicadas en el enunciado de la siguiente forma:

```
/*
Nombre de la función
Descripción de la función
-----------------------------------
Inputs:
(Tipo de dato) Descripción 1
(Tipo de dato) Descripción 2
.....
-----------------------------------
Output:
(Tipo de dato) Descripción

*/
```

8. Calificación

- Código
  - Orden (5 puntos)
  - TDA lista (35 puntos)
  - Funciones (60 puntos)
- No entrega Makefile (-100 puntos)
- No compila (-100 puntos)
- No utiliza punteros a void (-100 puntos)
- No comentar (-10 puntos c/u)
- No sigue las estructuras y definiciones propuestas (-20 puntos por cada error)
- Errores en uso de memoria (-20 MAX)
- Warnings de complilacion (-5 por cada uno)