INF-253 Lenguajes de Programación
Tarea 2: C

Profesor: José Luis Martí
Ayudante Cátedras: Juan Pablo Escalona
Ayudante Tareas: Cristian Vallejos

5 de abril de 2017

1. SansaBar

SansaBar es un juego que simula el comportamiento de un bar, y se ha solicitado a los estudiantes que cursan Lenguajes de Programación que programen la base de este nuevo simulador, utilizando **C** como lenguaje fuente, enfatizando en **punteros a void** y **punteros a funciones**.

El jugador podrá seleccionar la cantidad de garzones y barmans que desea contratar, y las mesas y decoraciones que comprará para aumentar la popularidad de su local y satisfacer a sus clientes.

2. Reglas del juego

- El bar atiende 15 horas al día.
- La cantidad de gente que llega al local o Clientes Potenciales (C_P) depende de la popularidad del bar.
$$C_P = 2 * 15 * Popularidad$$
- El bar tendrá mesas, las cuales admiten un máximo de 6 personas sentadas por hora. Si llegan más personas que las que se pueden sentar, se retirarán insatisfechas.
- La cantidad de Clientes Sentados en el día (C_S) está dada por la expresión:
$$C_S = min(C_P, 6 * 15 * Mesas)$$
- Cada garzón contratado es capaz de atender 3 mesas por hora.
- Así, la cantidad de Clientes Atendidos durante el día (C_A) está dada por:
$$C_A = min(C_S, 18 * 15 * Garzones)$$
- Cada barman contratado es capaz de servir 9 tragos por hora.
- Luego, la cantidad de Clientes Felices durante el día (C_F), debido a que fueron atendidos a tiempo, está dada por:
$$C_F = min(C_A, 9 * 15 * Barmans)$$
- La popularidad base del local es 1, puesto que no ha obtenido estrellas. Para aumentar las estrellas del bar se debe satisfacer a la mayoría de los clientes potenciales.

- Si C_F es mayor al 75 % de C_P durante una cierta cantidad de días seguidos, el bar aumentará en uno su número de estrellas (*según lo expuesto en la siguiente tabla*). La popularidad base se modifica según el número de estrellas.

Estrellas	Popularidad Base	Dias Seguidos
1	1	5
2	2	10
3	4	15
4	8	20
5	16	25

- Una vez obtenido cierto nivel de estrellas, no se puede bajar.
- Adicionalmente, la popularidad del local también puede verse aumentada con las decoraciones que el jugador compre.
- Para esto, existen 3 tipos de Decoraciones:
  - Regular: Agregar 0,25 a la popularidad.
  - Bonita: Agrega 0,5 a la popularidad.
  - Espectacular: Agrega 1 a la popularidad.
- Los tragos del local puede ser mejorados de nivel, lo que aumenta su Precio y Costo de preparación.
- El costo de las distintas decoraciones y las mesas debe ser definido por los programadores.
- El sueldo diario de los empleados (garzones y barmans) debe ser definido por los programadores.
- El Precio de los tragos, su costo de fabricación, y la función de crecimiento por nivel de éstos también deben ser definidos por los programadores.
- Sólo los Clientes Felices pagan lo preparado. Debido a esto, aquellos clientes que fueron atendidos pero no salen felices sólo generan costo para el dueño del bar.

$$Ingreso = C_F * PrecioTragos$$

$$Costo = C_A * CostoTragos + Sueldos$$

$$GanaciaoPerdida = Ingresos - Costos$$

- El jugador inicia con un Capital inicial (*definido por los programadores*). Si dada una pérdida se llega a un Capital total menor o igual a 0, el juego se acaba y el jugador pierde.
- El jugador puede vender decoraciones o mesas y despedir a los barmans o a los garzones.

3. Requerimientos

- Se debe implementar un programa que permita jugar el simulador descrito en la sección anterior.
- Se jugará mediante línea de comando con menús para realizar las compras, de modo que el jugador deberá ingresar caracteres para hacer selecciones.

- El Bar será una estructura que poseerá una lista de Empleados, una lista de Muebles, su Popularidad y su Capital total.
- Los empleados son estructuras que poseen su tipo y una función Pagar, la cual recibe el local y resta, del capital total, el dinero correspondiente al sueldo de ese empleado. Los empleados no deben tener su sueldo en la estructura, éste se encuentra en la función correspondiente. Además, los sueldos de los garzones y barmans no puede ser igual.
- Según el punto anterior, deben existir dos funciones *pagarGarzon* y *pagarBarman*, las cuales son asignadas al puntero a función en la estructura empleado correspondiente.
- Para realizar el pago de sueldos del día se debe recorrer la lista de empleados llamando a la función Pagar de cada uno de éstos.
- Para obtener el número total de garzones y barmans se debe recorrer toda la lista al inicio de cada día.
- Los muebles pueden ser decoraciones o mesas, poseen un tipo, un precio de compra, precio de venta y una popularidad. Las mesas no aportan a la popularidad.
- Para obtener el número de mesas se debe recorrer la lista contando todas las mesas.
- Para obtener el bono de popularidad de decoraciones, se debe recorrer toda la lista de muebles, sumando el valor de popularidad de cada uno de éstos a la popularidad base.
- Ambas listas deben ser implementadas por los programadores, utilizando la misma estructura de nodo, con un puntero a void al elemento.

4. Estructuras Básicas

```
struct Local{
    int dinerototal;
    int estrellas = 0;
    Lista empleados;
    Lista muebles;  }

struct Empleado{
    char tipo; //Barman o Garzon
    void(*pagar)(void *);
    //El parametro entregado es el local.
}

struct Mueble{
    char tipo; //Decoracion o Mesa
    double popularidad;
    int compra;
    int venta;  }
```

5. Archivos a Entregar

En resumen, los archivos mínimos a entregar serian:

- Empleado.c (*funciones de configuración de Empleado*).

- Empleado.h (*Archivo con estructuras y prototipos de funciones de Empleado*).
- Mueble.c (*funciones de configuración de Mueble*).
- Mueble.h (*Archivo con estructuras y prototipos de funciones de Mueble*).
- Local.c (*funciones de configuración de Local*).
- Local.h (*Archivo con estructuras y prototipos de funciones de Local*).
- main.c (*Archivo con programa principal*).
- MAKEFILE (*Archivo de compilación automática*).
- README.txt (*Archivo manual*).
- lista.c (*Archivo con funciones de lista enlazada*).
- lista.h (*Archivo con estructuras y prototipos de funciones de lista enlazada*).

Los estudiantes pueden crear más archivos si lo consideran necesario, mientras los mencionados anteriormente estén entre los enviados.

6. Sobre Entrega

- La revisión se efectuará sobre el sistema operativo presente en los computadores del laboratorio de la Universidad (*LDS*).
- El código debe venir identado y sin warnings.
- Cada función debe llevar una descripción según lo establecido por el siguiente ejemplo:

```
/******** Funcion: Suma_Enteros ******************************
Descripcion: suma dos enteros positivos
Parametros:
n1 entero
n2 entero
Retorno: resultado de la operacion aritmetica de la suma entero
*************************************************************/
```

- Debe estar presente el archivo MAKEFILE para que se efectúe la revisión.
- **La entrega debe realizarse en tarball (tar.gz) y debe llevar el nombre: Tarea2LP_RolIntegrante-1_RolIntegrante-2.tar.gz**
- **El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones detalladas para la compilación y utilización de su programa (*en caso de ser necesario*).**
- **El no cumplir con las reglas de entrega conllevará nota 0 en su tarea.**
- El plazo máximo de entrega es hasta el viernes 21 de abril a las 23:55 (*vía moodle*).
- Por cada día de atraso o fracción se descontarán 20 puntos.
- Las copias serán evaluadas con nota 0.

7. Calificación

La escala a utilizar para la revisión de la tarea es la siguiente:

- Código (80 puntos)
  - Orden (5 puntos)
  - Implementación (20 puntos)
  - Punteros Void (10 puntos)
  - Punteros Funciones (10 puntos)
  - Listas (20 puntos)
  - Estructuras (10 puntos)
  - Comentarios (5 puntos)
- Juego (20 puntos)
  - Cumple Reglas del Juego (20 puntos)

Los descuentos que pueden ocurrir son:

- No Compila (100 puntos)
- No Cumplir Reglas de Entrega (100 puntos)
- Warnings (5 puntos cada uno)