Tarea 1 - ELO320 Estructura de Datos y Algoritmos
Prof.: Nicolás Gálvez R.
Ayudantes: Raimundo Gross - Cristóbal Nilo - Matías Soto
Campus San Joaquín, UTFSM. 22 de Abril de 2022

1 PunteroTicket trae a Chile a: Conejo !Bueno

La aclamada productora y vendedora de entradas a eventos de *Ciudad C*, **PunteroTicket**, está organizando una gira por distintas ciudades del país del artista internacional del momento: **Conejo !Bueno**. Antes de iniciar la venta de entradas, la empresa quiere implementar un sistema de ventas online que incluya el manejo de una fila virtual aleatoria para los interesados, contratándolo a ud. para llevar a cabo dicha labor. La primera etapa consiste en la creación del núcleo de la plataforma para la venta de entradas considerando los siguientes elementos:

- La gira contempla tres (3) ciudades, cada una de las cuales posee un estadio con características diferentes para alojar el show.
- La venta de entradas se realizará de forma separada para cada show.
- La fila de posible asistentes será simulada, y considerará sólo a aquellos que se encuentren al momento de la apertura de la venta de entradas.
- Las personas sólo podrán adquirir entradas de un mismo sector y con una cantidad máxima de dos (2) unidades.

Los características de las locaciones que cobijarán los shows estarán almacenadas en archivos de texto plano cuyo nombre tendrá el formato `estadioX.txt`, donde `X` es el número del show a ser vendido. Por ejemplo, en `estadio1.txt` se encuentra la información de capacidad, ubicaciones disponibles y precios para el primer show a realizarse. Este archivo tiene un formato definido, tal como se muestra a continuación:

`estadio1.txt`

```
n_sec 5 precio 50000
    1 2500 5.0
    2 8500 3.8
    3 12500 2.5
    4 25000 1.2
    5 20000 0.8
```

En cada uno de estos archivos, la primera línea representa la información base: número de secciones existentes, y el precio base de las entradas para asistir al concierto (en pesos). Luego, habrá una línea por cada sección existente que incluirá: el número de la sección, la cantidad de entradas disponibles, y el factor de la entrada respecto al precio base. Así, `estadio1.txt` nos indica que dicha locación tiene cinco (5) secciones y el precio base de sus entradas es de $50000 pesos, para luego especificar que, por ejemplo, en la Sección 1 habrá 2500 entradas disponibles con un precio de $5.0 \times 50000 = \$250000$ pesos cada una.

La fila de clientes para la compra de entradas se hará de forma simulada y aleatoria. Este proceso debe cumplir con los siguientes requisitos:

- El número de posibles clientes será determinado al azar entre cinco (5) a veinte (20) veces la capacidad del estadio donde se realizará el show.
- Cada cliente tendrá un Identificador único (RUT, RUN, DNI, Rol, Id Correlativo, etc.), un presupuesto asociado, y la cantidad de entradas obtenidas para una locación específica.
- El presupuesto de cada cliente será designado de forma aleatoria entre la mitad del precio del sector más barato y tres veces el precio del sector más caro.

- Cada cliente sólo podrá comprar entradas de la sección más cara a la que pueda acceder, siempre respetando las reglas descritas en un inicio. Si no tiene presupuesto suficiente, no podrá adquirir entradas.

Finalmente, el sistema deberá generar dos tipos de reportes respecto al proceso de ventas, uno público y otro privado. El reporte público deberá ser desplegado por pantalla cada vez que se venda un múltiplo de 5000 entradas. Este reporte debe incluír:

- El número total de entradas disponibles.
- Localidades agotadas.
- (Opcional) Mensajes de sponsors, de referencias al artista principal, etc.

Por otra parte, los reportes privados se generarán en archivos de texto plano, cuyo nombre de formato será `<porcentaje>.txt` cada vez que se venda un cuarto de las entradas totales, vale decir, al vender 25%, 50%, 75% y 100% de las entradas. Por ejemplo, al momento de vender 50% de las entradas totales, se debe generar el reporte con nombre `50.txt`. Cada uno de estos archivos, debe contener la siguiente información:

- Número de entradas vendidas y restantes, además de la recaudación, por cada sección del estadio.
- Cantidad de clientes que no pudieron adquirir una entrada.
- Cantidad de clientes restantes en la fila simulada.
- Recaudación total en dicho instante.

Su superior directo, Cisco Heads, es experto en Estructura de Datos y Algoritmos, y le comenta que la dinámica para resolver el problema es la siguiente:

- Se debe leer obtener la información base de las características del recinto del show.
- Luego, se inicializa la fila aleatoria de compradores con todos los interesados.
- Posteriormente, se procesar la venta de cada uno de los clientes en el orden asignado, si fuese posible.
- Finalmente, y durante el transcurso de la venta general, se emiten los reportes correspondientes.

Además, le entrega el siguiente header (`.h`) con definiciones obligatorias a usar para hacer el núcleo de la plataforma. Su superior le da la opción de agregar cosas si ud. estima conveniente, pero **no puede modificar y debe usar** todo lo que se encuentra ya definido.

`entradas.h`

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

const double pct_interno = 0.25;
const int cant_externo = 5000;

typedef struct seccion{
    int id;
    int asientos;
    float factor;
} sector;

typedef struct estadio{
    int cant_sec;
    int pbase;
    sector *secc;
} stadium;

typedef struct cliente{
    char *id;
    int presupuesto;
    int n_seccion;
    int n_entradas;
    struct cliente *sgte;
} fans;
```

Su jefe cree que la labor es clara, y utilizando los conceptos de su formación académica en ELO-320 Estructura de Datos y Algoritmos, le solicita implementar la plataforma descrita, y simular la venta de entradas para un show determinado. Ud. específicamente debe:

(a) (10 puntos) Leer el archivo del recinto del show del cual se venderán entradas, (por ejemplo, `estadio1.txt`), y almacenar su información en una Estructura de Datos de su elección. Justifique en su código el por qué de su elección. **Considere que el nombre del archivo será obligatoriamente un parámetro del ejecutable**.

(b) (15 puntos) Inicializar la lista de clientes para la venta de entradas, utilizando una estructura de datos de su elección. Justifique en el código, el por que de su elección. Recuerde que la ubicación de cada cliente en la Estructura de Datos, debe ser aleatoria, lo cual se puede realizar libremente y de variadas formas. Escoja una que le acomode y le haga sentido, además explíquela.

(c) (30 puntos) Crear un archivo, llamado `tda.c`, en donde implemente la interfaz de la estructura de datos escogida en los puntos anteriores. Diferencie bien, las funciones si usa dos o más estructuras diferentes. Si ud. desea puede crear más de un archivo `.c` por cada estructura de datos utilizada.

(d) (20 puntos) Crear una función para la venta de entradas, que utilizando la estructura de datos seleccionada para la simulación de la fila, vaya pasando por cada cliente, verificando si cumple los requisitos explicados en el enunciado para acceder a una entrada para el show. Tome en consideración, que clasificar a los clientes post-venta, puede ser una buena estrategia de cara a los reportes.

(e) (15 puntos) Crear una función capaz de generar los reportes privados solicitados, a través de archivos de texto plano, durante la venta de entradas.

(f) (10 puntos) Crear una función capaz de generar los reportes públicos solicitados, por salida estándar, durante la venta de entradas.

(g) (10 puntos) Bono: Crear una función capaz de generar mensajes especiales a aparecer durante la venta de entradas. Éstos deben tener relación con los sponsor del espectáculo, referencias al artista, situaciones de las entradas, etc. Sea creativo.

2 Reglas de entrega y Consideraciones Generales

- Este trabajo debe realizarse **individualmente**, vale decir, en **grupos de un (1) estudiante**. No se harán excepciones.
- El programa debe ser desarrollado en lenguaje C, y compilado con la versión de gcc disponible en el servidor Aragorn[1]: `gcc 4.8.5`.
- No hay un límite máximo de funciones a realizar, si ud. desea modularizar al máximo, tiene la libertad de hacerlo.
- La tarea debe ser entregada en la plataforma **AULA USM**, el día Viernes 27 de Mayo de 2022 hasta las 23:59:59 Hora de Chile Continental (UTC -4).
- Cada minuto de atraso, implicará un descuento siguiendo la sucesión de Fibonacci. Así, diez minutos de atraso implicarían un descuento de cincuenta y cinco (55) puntos ($f(10) = 55$).
- La tarea debe incluirse en un archivo comprimido **.tar.gz**. El nombre del archivo debe seguir la siguiente estructura: `tarea1-eda-nombre-apellido-paralelo.tar.gz`; e.g., `tarea1-eda-oliver-atom-p200.tar.gz`.
- El archivo comprimido debe incluir lo siguiente:
  - Cabecera `.h`: Archivo cabecera en el cual se deben incluir todas las bibliotecas a usar en el programa, además de las definiciones de macros, variables globales, tipos de datos personalizados, struct y prototipos de todas las funciones.

[1] `ssh aragorn.elo.utfsm.cl -l cuentausm`

  - Código `.c`: Código del programa solicitado. Considere que éste es un desafío de al menos tamaño mediano, por lo tanto es recomendable dividir el código en diferentes archivos agrupados por su finalidad.
  - `README`: Archivo de texto plano en el cual se debe incluir: (1) una pequeña reseña del programa, (2) las condiciones de compilación y ejecución, (3) las instrucciones de compilación y ejecución, y (4) la información del creador del sistema.
  - `Makefile`: Archivo de compilación automática del sistema. Una cápsula de video para su confección se encontrará disponible en AULA USM.

- La revisión de los programas se hará utilizando diferentes archivos de tamaño variable. Por lo tanto, debe programar de forma genérica para cualquier cantidad de información de entrada.
- Cada fuga de memoria será penalizada. Utilice `valgrind` para verificar la correcta asignación, uso y liberación de memoria en la ejecución de su programa.
- Si su programa no compila, su nota será automáticamente **un cero (0)**.
- Cualquier atisbo de copia, será penalizada con máxima severidad.
- No deje la tarea para último momento, planifique bien sus tiempos y constrúyala paso a paso.
- Consulte sus dudas, lo más pronto posible, a través de los diversos medios de la asignatura.