INF-134 Estructuras de Datos, 2022-1

Tarea 3

Profesores: Gabriel Carmona, Javier Robledo
gabriel.carmonat@sansano.usm.cl, javier.robledo@usm.cl

Ayudantes:
Joaquín Gatica (joaquin.gatica@sansano.usm.cl)
Rodrigo Flores (rodrigo.floresf@sansano.usm.cl)
Gabriel Escalona (gabriel.escalona@usm.cl)

Fecha de entrega: 30 de noviembre, 2022.
Plazo máximo de entrega: 5 días.

1. Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes (usando los correos indicados en el encabezado de esta tarea). No se permiten de ninguna manera grupos de más de 3 personas. Debe usarse el lenguaje de programación C++. Al evaluarlas, las tareas serán compiladas usando el compilador `g++`, usando la línea de comando `g++ archivo.cpp -o output -Wall`. **No se aceptarán variantes o implementaciones particulares de `g++`, como el usado por MinGW**. Se deben seguir los tutoriales que están en Aula USM, cualquier alternativa explicada allí es válida. Recordar que una única tarea en el semestre puede tener nota menor a 30. El no cumplimiento de esta regla implica reprobar el curso. No se permite usar la biblioteca *stl*, así como ninguno de los *containers* y algoritmos definidos en ella (e.g. `vector`, `list`, etc.). Está permitido usar otras funciones de utilidad definidas en bibliotecas estándar, como por ejemplo `math.h`, `string`, `fstream`, `iostream`, etc.

2. Objetivos

Entender y familiarizarse con la implementación y utilización de estructuras de datos tipo hashing.

Además se fomentará el uso de las buenas prácticas y el orden en la programación de los problemas correspondientes.

3. Problema

Un profesor anónimo es súper fanático de los juegos de cartas, entonces el decide crear su propio juego de cartas llamado *USM The Informatiring*. Para hacer su juego de cartas, decide abrir una tienda de cartas dentro de la universidad. En esta tienda, el profesor anónimo vende cartas individuales y sobres de 10 cartas.

Por coincidencias, el profesor dicta Estructuras de Datos, por lo tanto él sabe que utilizar la técnica de hashing permite saber que cartas individuales y sobres tiene en ese momento en tiempo eficiente. Por lo tanto, se le ocurre pedirle a sus alumnos que realicen un código que permita manejar la información de su tienda.

3.1. Cartas y Sobres

Para efectos del problema las cartas se representarán por el siguiente struct:

```
struct Carta {
    int id;
    string nombre;
    int ataque;
    int defensa;
    int precio;
}
```

Y los sobres se representarán por el siguiente struct:

```
struct Sobre {
    int id;
    Carta cartas[10];
}
```

Debe implementar dos tablas de hashing. Una tabla almacenará las cartas y la cantidad de esa misma. La otra tabla almacenará los sobres que hay.

3.2. Funcionalidades

- `int tengo_la_carta(int id)`: debe retornar la cantidad de unidades que quedan de la carta con la respectiva `id`.
- `void mostrar_cartas()`: debe mostrar por pantalla todas las cartas que hay en la tienda hasta ese momento. Se mostrará cada carta con el siguiente formato:

  `id nombre ataque defensa precio cantidad`
- `void mostrar_sobres()`: debe mostrar por pantalla todos los sobres que hay en la tienda con el siguiente formato:

  `id`
- `void vender_carta(int id)`: debe vender la carta con el `id` respectivo, por lo que se debe reducir la cantidad de esa carta en 1 y si la cantidad de esa carta llega a 0 se debe eliminar de la tabla de hashing. Finalmente, debe sumar el precio al dinero recolectado hasta ese momento. Si se pudo vender la carta se debe mostrar por consola `Vendida la carta!`. Si el `id` no se encuentra en la tabla, se debe mostrar por consola `ese id no se encuentra`
- `void vender_sobre(int id)`: debe vender el sobre con el `id` respectivo y mostrar su contenido por pantalla. Luego, debe eliminar ese sobre de la tabla de hashing y sumar 1000 al dinero recolectado hasta ese momento. Si se pudo vender el sobre se debe mostrar por consola `Vendido el sobre!`. Si el `id` no se encuentra en la tabla, debe mostrar por consola `Ese id no se encuentra!`

3.3. Programa

Su programa debe primero leer un archivo de texto llamado `Tienda.txt`, el cual contiene la siguiente información:

- Un entero n, correspondiente a la cantidad de cartas individuales en la tienda.
- Luego, le siguen n líneas, dónde cada línea corresponde a la descripción de una carta con el siguiente formato:

  ```
  id nombre ataque defensa precio
  ```

- Luego, se verá un entero m, correspondiente a la cantidad de sobres en la tienda.
- Después del entero, le siguen $11 \cdot m$ líneas, dónde la primera línea corresponde a la *id* del sobre y las siguientes 10 líneas corresponde a las cartas en ese sobre.

Luego, por consola se ingresará un entero o, el cuál corresponderá a una operación a realizar, las posibles operaciones son:

- Si $o = 1$, entonces se debe mostrar por consola el dinero recolectado hasta el momento
- Si $o = 2$, entonces se debe mostrar por consola las cartas individuales en la tienda hasta el momento
- Si $o = 3$, entonces se debe mostrar por consola los sobres en la tienda hasta el momento.
- Si $o = 4$, entonces se debe vender una carta, ingresando otro entero *id* correspondiente al *id* de la carta.
- Si $o = 5$, entonces se debe vender un sobre, ingresando otro entero *id* correspondiente al *id* del sobre.
- Si $o = 6$, entonces se debe terminar el programa

3.4. Ejemplo

Se tiene el siguiente contenido en el archivo `Tienda.txt`:

```
5
10 El_Destructor_Demoniaco 7 3 500
12 Asesino_Discreto 6 8 750
17 Ma_tetres 1 2 1000
10 El_Destructor_Demoniaco 7 3 500
8 Locos_Puntuales 9 9 1500
3
12376
10 El_Destructor_Demoniaco 7 3 500
12 Asesino_Discreto 6 8 750
17 Ma_tetres 1 2 1000
10 El_Destructor_Demoniaco 7 3 500
8 Locos_Puntuales 9 9 1500
10 El_Destructor_Demoniaco 7 3 500
12 Asesino_Discreto 6 8 750
17 Ma_tetres 1 2 1000
10 El_Destructor_Demoniaco 7 3 500
8 Locos_Puntuales 9 9 1500
101010
12 Asesino_Discreto 6 8 750
```

```
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
1231845
10 El_Destructor_Demoniaco 7 3 500
12 Asesino_Discreto 6 8 750
17 Ma_tetres 1 2 1000
10 El_Destructor_Demoniaco 7 3 500
8 Locos_Puntuales 9 9 1500
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
```

Luego por consola sucederá lo siguiente:

```
2
10 El_Destructor_Demoniaco 7 3 500 2
12 Asesino_Discreto 6 8 750 1
17 Ma_tetres 1 2 1000 1
8 Locos_Puntuales 9 9 1500 1
3
12376
101010
1231845
4 12
Vendida la carta!
4 12
Ese id no se encuentra!
4 10
Vendida la carta!
2
10 El_Destructor_Demoniaco 7 3 500 1
17 Ma_tetres 1 2 1000 1
8 Locos_Puntuales 9 9 1500 1
5 101010
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
12 Asesino_Discreto 6 8 750
```

```
12 Asesino_Discreto 6 8 750
Vendido el sobre!
5 101010
Ese id no se encuentra!
3
12376
1231845
1
2250
6
```

Nota: en el ejemplo las letras en itálica significa que son input y las que no significa que son output

4. Consideraciones

- Se asume que todos los datos se ingresarán correctamente.
- Si dos cartas tienen el mismo id, entonces tienen los mismos datos.
- Parte de la tarea, es que usted programe funciones de hashing y que resuelva colisiones para almacenar los elementos descritos anteriormente.
- Su función de hashing y la resolución de colisiones no deben ser las funciones triviales, eso quiere decir $h(k) = k$ y $p(k, i) = i$.
- Su tabla de hashing debe cumplir con un factor α de 0,6.
- La máxima cantidad de cartas y sobres que se ingresará será de 1000.
- El orden en que se imprimen las cartas o sobres puede ser cualquiera, con tal de mostrar la información correcta.

5. Entrega de la Tarea

Entregue la tarea enviando un archivo comprimido `tarea3-apellido1-apellido2-apellido3.zip` o `tarea3-apellido1-apellido2-apellido3.tar.gz` (reemplazando sus apellidos según corresponda) a la página aula.usm del curso, a más tardar el día 30 de noviembre, 2022, a las 23:59:00 hs (Chile Continental), el cual contenga como mínimo:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. ¡Los archivos deben compilar!
- **nombres.txt**, Nombre, ROL, Paralelo y qué programó cada integrante del grupo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias, y la forma de compilación que usó para cada problema de la tarea (debe ser alguna de las indicadas en los tutoriales entregados en Aula USM).

6. Restricciones y Consideraciones

- Por cada día de atraso en la entrega de la tarea se descontarán 10 puntos en la nota.
- El plazo máximo de entrega es 5 días después de la fecha original de entrega.

- **Las tareas que no compilen no serán revisadas y serán calificadas con nota 0**.
- Debe usar **obligatoriamente** alguna de las formas de compilación indicadas en los tutoriales entregados en Aula USM.
- Por cada *Warning* en la compilación se descontarán 5 puntos.
- Si se detecta **COPIA** la nota automáticamente sera 0 (CERO), para todos los grupos involucrados. El incidente será reportado al jefe de carrera.
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.

7. Consejos de Programación

El código fuente del programa debe estar estructurado adecuadamente en archivos (separados de ser necesario). Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota.

Cada función programada debe tener comentarios de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

Por cada comentario faltante, se restarán 5 puntos.

Por último, la indentación (1 TAB o 4 espacios), es muy importante. Por **cada bloque mal indentado, se quitarán 10 puntos**.