INF-134 Estructuras de Datos, 2022-1

Tarea 2

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: 04 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 listas y árboles.

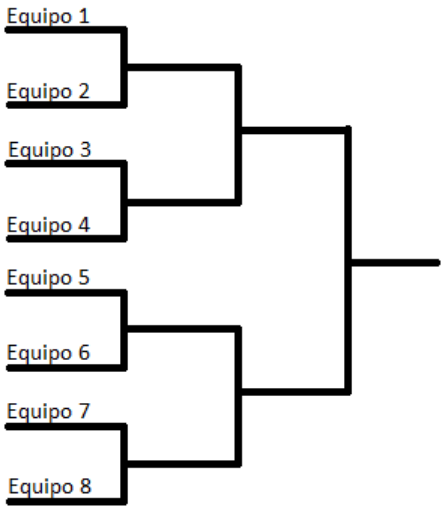
Además se fomentará el uso de las buenas prácticas y el orden en la programación de los problemas correspondientes.

3. Problema

Los juegos en línea se están haciendo cada vez más populares. Por lo tanto, organizar torneos es importante. Un torneo de eliminación directa representado por *Brackets* son los torneos más comunes.

Este tipo de torneos consiste en una cantidad de equipos que se enfrentan en partidos uno contra uno. De cada partido sale un ganador y perdedor. El ganador avanza de ronda mientras el perdedor no.

Por ejemplo, en la siguiente figura tenemos un torneo recién iniciado de este tipo el cual contiene 8 equipos.



Luego, por cada ronda avanzada se resolverán los partidos hasta quedar de la siguiente forma:

Y por lo tanto representarlos es un problema interesante a resolver. Para esta tarea este torneo corresponderá a un Arbol Binario y los participantes serán equipos que corresponderán a una Lista Enlazada.

Ambas estructuras corresponden a las vistas en clases.

3.1. Equipo

La Lista Enlazada en este caso será llamada `Equipo`. Los elementos de esta serán representados por el siguiente struct:

```
struct Persona {
    string nombre;
    bool capitan;
    int poder;
}
```

Esta clase deberá implementar las siguientes operaciones:

- `int agregar_companero(String nombre, int poder)`: debe recibir un nombre y el poder de una persona e insertarla al equipo. Debe retornar la posición en la cual la persona fue insertada.
- `int calcular_poder()`: debe retornar la suma de los poderes de cada Persona.
- `void imprimir_equipo()`: debe imprimir el Equipo con el siguiente formato:

  ```
  Equipo:
  Persona 0: nombre capitan poder
  Persona 1: nombre capitan poder
  .
  .
  .
  Persona n - 1: nombre capitan poder
  ```

Nota: debe implementar toda operación que sea importante para las listas enlazadas.

3.2. Torneo

El Árbol Binario en este caso será llamada `Torneo`. Los elementos de este serán de tipo `Equipo`.
Esta clase deberá implementar las siguientes operaciones:

- `void crear_torneo(Equipo* equipos, int n)`: debe recibir un puntero a `Equipo` y la cantidad de equipos e inicializar los nodos del árbol con los equipos en el mismo orden.
- `void imprimir_bracket()`: debe imprimir el estado actual del torneo, dónde por cada equipo que deba imprimir solo debe imprimir el nombre del capitán y el poder total del equipo.
- `void avanzar_ronda()`: debe avanzar una ronda del torneo.

Nota: debe implementar toda operación que usted encuentre importante para el funcionamiento de un árbol binario.

3.3. Funciones

A parte de las dos clases previamente mencionadas, debe implementar la siguiente función:

- `bool batallar(Equipo &a, Equipo &b)`: debe recibir dos equipos. Debe retornar `true` si el equipo a tiene más poder y `false` si el equipo b tiene más poder.

3.4. Tarea de su código

Su código debe ser capaz de leer una cantidad n de equipos e iniciar un torneo. Luego, por consola habrán tres tipos de consultas: avanzar ronda, imprimir bracket actual e imprimir algún equipo.

La entrada consiste primero de un archivo de texto llamado `Equipos.txt`. El cual tendrá primero un entero n, que corresponde a la cantidad de equipos a leer. Luego, le siguen n equipos los cuales tendrán la siguiente descripción:

- Un entero p correspondiente a la cantidad de personas en el equipo.
- Luego le siguen p líneas que contienen el nombre de la persona y su poder.
- Por último, hay una línea que contiene un nombre que corresponde al capitán del equipo.

Segundo, las consultas serán realizadas por entrada estándar. Por cada línea de esta hay un entero t, el cual puede significar lo siguiente dependiendo de su valor:

- Si $t = 1$, entonces se debe avanzar una ronda en el bracket.
- Si $t = 2$, entonces se debe imprimir el estado actual del bracket.
- Si $t = 3$, entonces en la misma línea se verá un entero k y se debe imprimir a todos los equipos que tengan poder mayor o igual a k.

Cuando no sea posible más avanzar de ronda se debe imprimir el bracket final y terminar el programa.

3.5. Ejemplo

Se tiene el siguiente Equipos.txt:

```
4
4
Mario 20
Luigi 10
Peach 15
Daisy 17
Mario
3
Sonic 25
Shadow 10
Tails 15
Sonic
2
Pikachu 30
Ash 20
Ash
1
FLUDD 60
FLUDD
```

Luego por consola sucederá lo siguiente:

```
2
--
-- vs --
```

```
Mario 62 vs Sonic 50 | Ash 50 vs FLUDD 60
1
2
--
Mario 62 vs FLUDD 60
Mario 62 vs Sonic 50 | Ash 50 vs FLUDD 60
3 60
Equipo:
Persona 0: Mario true 20
Persona 1: Luigi false 10
Persona 2: Peach false 15
Persona 3: Daisy false 17
Equipo:
Persona 0: FLUDD true 60
1
Mario 62
Mario 62 vs FLUDD 60
Mario 62 vs Sonic 50 | Ash 50 vs FLUDD 60
```

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

4. Consideraciones

- La cantidad de equipos siempre serán equivalentes a 2^k.
- Si dos equipos tienen el mismo poder, siempre se elige al equipo de la izquierda como ganador.
- Se deben entregar dos archivos para Equipo: `Equipo.hpp` y `Equipo.cpp`.
- Se deben entregar dos archivos para Torneo: `Torneo.hpp` y `Torneo.cpp`.
- Se debe entregar un archivo `Main.cpp`, el cual ejecutará el código.
- Pueden agregar cualquier operación a las clases y cualquier función que ustedes estimen conveniente.

5. Entrega de la Tarea

Entregue la tarea enviando un archivo comprimido `tarea1-apellido1-apellido2-apellido3.zip` o `tarea1-apellido1-apellido2-apellido3.tar.gz` (reemplazando sus apellidos según corresponda) a la página aula.usm del curso, a más tardar el día 04 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**.