Explicación del código

Nombre: Aldo Ray Vasquez Lopez

Fecha: 03/11/2023

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

vector<int> Torneo(vector<int> habilidades, int N, int K) {}

int main() {}
```

El código se divide principalmente en dos funciones, nuestra función principal y la función Torneo. Así mismo, se hace uso de la librería <vector>, para almacenar las habilidades que se nos brindan, y la librería <queue>, la cual es fundamental para realizar las colas.

```
vector<int> Torneo(vector<int> habilidades, int N, int K) {
    int tam = habilidades.size();

    queue<int> torneoQueue;

    for (int i = 2; i < tam; i++) {
        torneoQueue.push(habilidades[i]);
    }

    int rondasGanadas = 0;
    int jugador1 = habilidades[0];
    int jugador2 = habilidades[1];

    vector<int> resultado(2);
```

Este parte del código establece la base para la simulación del torneo, inicializando la cola con los jugadores y preparando las variables para llevar un registro de las rondas ganadas y los resultados de los juegos.

- int tam = habilidades.size(); calcula el tamaño del vector habilidades, lo que determina la cantidad total de jugadores en el torneo.
- queue<int> torneoQueue; crea una cola (FIFO) llamada torneoQueue para almacenar a los jugadores que participarán en el torneo. Inicialmente, está vacía
- El bucle for recorre el vector habilidades desde el tercer elemento (índice 2) hasta el final del vector. Durante cada iteración, agrega el valor de habilidades del jugador en la cola torneoQueue. Esto llena la cola con todos los jugadores, excepto los dos primeros (jugador1 y jugador2).
- int rondasGanadas = 0; inicializa la variable rondasGanadas a 0, que llevará un registro del número de rondas ganadas por un jugador en un juego.

- int jugador1 = habilidades[0]; y int jugador2 = habilidades[1]; inicializan las habilidades de los dos primeros jugadores con los valores del vector habilidades. Estos jugadores serán los primeros en competir en el torneo.
- vector<int> resultado(2); crea un vector llamado resultado que almacenará los valores de las habilidades de los jugadores ganadores de cada juego. Está inicializado con dos elementos, pero se sobrescribirá con los valores adecuados en cada juego.

```
    while (K > 0) {
        if (jugador1 > jugador2) {
            rondasGanadas++;
            resultado = {jugador2, jugador1};
            torneoQueue.push(jugador2);
        } else {
            rondasGanadas = 1;
            resultado = {jugador1, jugador2};
            torneoQueue.push(jugador1);
            jugador1 = jugador2;
        }

        if (rondasGanadas == N) {
            torneoQueue.push(jugador1);
            jugador1 = torneoQueue.front();
            torneoQueue.pop();
            rondasGanadas = 0;
        }
        jugador2 = torneoQueue.front();
        torneoQueue.pop();

        K--;
    }

    return resultado;
}
```

Aquí se implementa la simulación del torneo y la lógica para llevar un registro de las rondas ganadas, los resultados de los juegos y reducir el valor de K.

1. while (K > 0) { Inicia un bucle while que se ejecutará mientras K (el número total de juegos a jugar) sea mayor que cero. El bucle controla la simulación de los juegos del torneo.
2. if (jugador1 > jugador2) { Comprueba si las habilidades del jugador1 son mayores que las del jugador2. Si es así, significa que el jugador1 gana el juego.
    - rondasGanadas++; Incrementa la variable rondasGanadas, que lleva un registro del número de rondas ganadas por el jugador1.
    - resultado = {jugador2, jugador1}; Actualiza el vector resultado con las habilidades de los jugadores ganadores (jugador2 y jugador1).
    - torneoQueue.push(jugador2); Agrega al jugador2 a la cola torneoQueue, ya que ganó el juego.

3. En el bloque else, se maneja la situación en la que el jugador2 gana el juego.
    - rondasGanadas = 1; Restablece rondasGanadas a 1, ya que el jugador2 ganó una ronda.

- resultado = {jugador1, jugador2}; Actualiza el vector resultado con las habilidades de los jugadores ganadores (jugador1 y jugador2).
- torneoQueue.push(jugador1); Agrega al jugador1 a la cola torneoQueue.
- jugador1 = jugador2; Actualiza las habilidades del jugador1 con las habilidades del jugador2 para el próximo juego.

4. if (rondasGanadas == N) { Comprueba si un jugador ha ganado el número deseado de rondas.
   - torneoQueue.push(jugador1); Agrega al jugador1 (ganador del juego) a la cola torneoQueue.
   - jugador1 = torneoQueue.front(); Actualiza las habilidades del jugador1 con las habilidades del próximo jugador en la cola.
   - torneoQueue.pop(); Elimina al jugador1 de la cola.
   - rondasGanadas = 0; Restablece las rondas ganadas a 0, ya que comienza un nuevo juego.
5. jugador2 = torneoQueue.front(); Actualiza las habilidades del jugador2 con las habilidades del próximo jugador en la cola.
6. torneoQueue.pop(); Elimina al jugador2 de la cola, ya que ha completado un juego.
7. K--; Reduce el valor de K en 1 para llevar un registro de los juegos restantes en el torneo.

```cpp
int main() {
    vector<int> habilidades1 = {1, 2, 3};
    vector<int> habilidades2 = {49, 24, 26, 12, 5, 33, 25, 30, 35, 41, 46, 23, 21, 3, 38, 43, 11, 19, 34, 29, 20, 3
    vector<int> habilidades3 = {30, 12};

    vector<int> resultado1 = simularTorneo(habilidades1, 2, 2);

    vector<int> resultado2 = simularTorneo(habilidades2, 10, 399);

    vector<int> resultado3 = simularTorneo(habilidades1, 2, 4);

    vector<int> resultado4 = simularTorneo(habilidades3, 34, 80);

    cout << "Caso 1: " << resultado1[0] << " " << resultado1[1] << endl;
    cout << "Caso 2: " << resultado2[0] << " " << resultado2[1] << endl;
    cout << "Caso 3: " << resultado3[0] << " " << resultado3[1] << endl;
    cout << "Caso 4: " << resultado4[0] << " " << resultado4[1] << endl;
    return 0;
}
```

Por último, en el main se implementa la función con los diferentes casos de prueba, los cuales han sido solicitados en el pdf del lab07.