Programozói verseny

Ebben a feladatban egy **programozói verseny rendszerét** kell modellezned két osztály segítségével:

- `Task`, ami egy versenyfeladatot ír le,
- `Contest`, ami egy teljes versenyt modellez, amely több feladatot tartalmaz.

Task osztály

A `Task` osztály egyetlen versenyfeladatot ír le. Minden feladatról a következő adatokat kell tárolni:

- azonosító (ID), ami automatikusan és inkrementálisan generálódik
- név, azaz a feladat neve, pl. “Legrövidebb utak”
- kategóriák (több is lehet), pl. “feszítőfa”, “dinamikus programozás”, stb.
- nehézség (1–10 közötti szám)
- pontszám, amit a feladat ér (pl. 100 pont)

A `Task` osztály feladatai:

- Az **ID** automatikusan generálódik minden új feladathoz.
- Legyen **copy constructor és assignment operator**, hogy a feladatok másolhatók legyenek.
- Két `Task` akkor tekinthető **azonosnak**, ha ugyanaz az ID-juk.
- Biztosítsunk módot arra, hogy a **nehézségi szintet kategorizálni** is tudjuk a pontszám szerint. Azaz szükségünk lesz egy olyan függvényre, ami a szint szöveges leírását visszatéríti:
  - pl. “*könnyű*”, ha a nehézségi foka 1-3 között van, “*közepes*”, ha 4-7, és “*nehéz*”, ha 8-10.
- A << operátorral lehessen szépen **kiírni** a feladatot, a következő sablon szerint, a feladatok között hagyjunk egy-egy új sort elválasztóként:

```
[ID: <id>] <feladat_neve>
Kategoria: <kategoria_1>, <kategoria_2>
Nehézség: <szam> (<nehezseg_kategoria>) | Pont: <pontszam>
```

egy konkrét példa:

```
[ID: 1] Legrovidebb utak
Kategoria: algoritmus, feszitofa
Nehézség: 7 (közepes) | Pont: 100
```

A `Contest` osztály

A `Contest` osztály a versenyt modellezi, amely több `Task`-ot tartalmaz.

A `Contest` osztály képes kell legyen a következőekre:

- Új feladat **hozzáadása**:
  - Egy új `Task` objektumot lehet a versenyhez adni
  - Ha már létezik azonos ID-jú feladat, kivételt dob.
- Feladat **törlése ID alapján**:
  - Ha nincs ilyen feladat, szintén kivétel dobódik.
- **Keresés kategória alapján**, ami visszaadja az adott kategóriába tartozó feladatokat.
- **Átlagos nehézség** kiszámítása: a versenyben szereplő feladatok átlagos nehézségét.
- **Összpontszám** kiszámítása.
- Legnehezebb feladat **keresése**:
  - Visszaad egy legmagasabb nehézségű feladatot.
  - Ha több van, akkor ezek közül bármelyiket visszatérítheti.
- **Rendezés** pontszám szerint: növekvő sorrendbe rendezi a feladatokat a pontszámuk alapján.
- Készíts **másoló konstruktort és értékadó operátort** (`operator=`), hogy egy verseny másolható legyen.
- Verseny **kiírása**: a << operátor segítségével lehessen a verseny összes feladatát kiírni.

main.cpp teszteléshez

```cpp
#include <iostream>
#include "Task.h"
#include "Contest.h"


int main()
{
    Contest c("ECN 2025");

    Task t1("Pizza", "dinamikus programozas, algoritmus", 7, 65);
    Task t2("Szamjegyek osszege", "algoritmus", 1, 10);
    Task t3("Legrovidebb utak", "algoritmus, feszitofa", 8, 80);
    Task t4("Graf bejaras", "algoritmus", 8, 80);
```

```
14      Task t5("Leghosszabb kozos szekvencia", "greedy", 4, 50);
15
16      c.addTask(t1);
17      c.addTask(t2);
18      c.addTask(t3);
19      c.addTask(t4);
20      c.addTask(t5);
21
22      std::cout << c;
23
24      std::cout << "--------------------------\n";
25      std::cout << "Legnehezebb feladat:\n"
26                << c.getHardestTask() << "\n";
27
28      std::cout << "--------------------------\n";
29      std::cout << "Atlagos nehezseg: " << c.averageDifficulty() << "\n\n
            ";
30      std::cout << "--------------------------\n";
31      std::cout << "Osszpontszám: " << c.totalPoints() << "\n\n";
32
33      std::cout << "--------------------------\n";
34      std::cout << "Algoritmus kategoria feladatai:\n";
35      auto algoTasks = c.searchByCategory("algoritmus");
36      for (const auto &t : algoTasks)
37          std::cout << t << "\n";
38
39      std::cout << "--------------------------\n";
40      std::cout << "Feladatok rendezese pontszám szerint\n";
41      c.sortByPoints();
42      std::cout << c << "\n";
43
44      std::cout << "--------------------------\n";
45      std::cout << "Feladat torlese (ID=2)\n";
46      c.removeTask(2);
47      std::cout << c << "\n";
48
49      std::cout << "Nem letezo feladat torlese\n";
50      try
51      {
52          c.removeTask(10);
53      }
54      catch (const std::exception &e)
55      {
56          std::cout << "Hiba elkapva: " << e.what() << "\n\n";
57      }
58
59      std::cout << "--------------------------\n";
60      std::cout << "Verseny masolasa...\n";
61      Contest copied = c;
62      std::cout << "Masolat tartalma:\n"
63                << copied << "\n";
```

```
64
65      return 0;
66  }
```