Výsledková listina

Termín odevzdání:	**08.05.2023 11:59:59**
Pozdní odevzdání s penalizací:	**21.05.2023 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**5.0000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	7 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	1 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je realizovat šablonu třídy, která dokáže sestavovat výsledkové listiny v závislosti na zadaných vstupech - výsledcích zápasů dvojic soupeřů.

Předpokládáme, že máme několik soupeřů. V jednom zápase se utká dvojice soupeřů, výsledek zápasu je nějak popsán (například booleovskou hodnotou, dvojicí počtu bodů, dvojicí časů, ...). Protože chceme mít třídu obecnou, bude výsledek zápasu reprezentován jako generický datový typ. Takto bude do třídy vloženo několik výsledků pro zadané dvojice soupeřů (soupeři jsou jednoznačně identifikováni jménem - řetězcem). Následně chceme zjistit, zda na základě zadaných výsledků lze sestavit jednoznačnou výsledkovou listinu a případně tuto listinu i vypočítat. Trikem je, že při sestavování výsledku chceme vycházet pouze z informací o vítězi/poraženém ze zadaných zápasů.

Výsledková listina je určena jednoznačně, pokud pro každého soupeře X_i na pozici i platí:

- X_i porazil všechny soupeře X_j, $i+1 \leq j \leq n$. Soupeře porazil buď přímo, nebo zprostředkovaně (X_i porazil X_a, X_a porazil X_b, ..., X_z porazil X_j, $i+1 \leq a < b < \ldots < j \leq n$),
- X_i nevyhrál ani neremizoval se žádným soupeřem X_k, $0 \leq k \leq i-1$.

Vezměme příklad, kdy bylo zadáno, že A porazí B a B porazí C. Pak je zřejmě správná výsledková listina `A B C`. Pokud by ale bylo zadáno, že:

```
A porazí B
B porazí C
C porazí A
```

případně:

```
A porazí B
A porazí C
```

případně:

```
A porazí B
B remizuje s C
```

pak výsledkovou listinu nelze jednoznačně určit (museli bychom dodat informace o dalších zápasech nebo pořadí stanovit i na základě dalších kritérií, to v této úloze nebudeme dělat).

Bylo zmíněno, že zápas je zadaný jako trojice (soupeř1, soupeř2, výsledek), kde výsledek je nějaký datový typ popisující například skóre, časy nebo cokoliv jiného. Výsledek je potřeba převést na informaci o vítězi/remíze/poraženém, to bude mít na starosti porovnávač (komparátor) předaný metodě pro určení výsledkové listiny. Porovnávač je funkce, funktor nebo lambda-výraz, který pro daný výsledek vrací návratovou hodnotu:

- zápornou, pokud soupeř1 prohrál,
- nulovou, pokud soupeř1 a soupeř2 remizovali,
- kladnou, pokud soupeř1 vyhrál.

Celkové rozhraní šablony třídy `CContest` bude:

šablona je parametrizovaná generickým parametrem `M_`
: tento typ popisuje výsledek zápasu. Pro datový typ `M_` je garantováno, že je přesouvatelný, kopírovatelný a zrušitelný (CopyConstructibe, MoveConstructible, CopyAssignable, MoveAssignable a Destructible). Další vlastnosti typu nejsou garantované, speciálně, není garantované, že typ `M_` má implicitní konstruktor (nemusí být DefaultConstructible).

implicitní konstruktor
: vytvoří prázdnou instanci `CContest`,

`addMatch(contestant1, contestant2, result)`
: metoda přidá výsledek mezi soupeřem `contestant1` a `contestant2`. Soupeři jsou zadaní v podobě řetězců. Výsledek `result` je datového typu `M_`. Metoda si uloží informaci o tomto zápasu do nějakého úložiště ve své instanci. Volání `addMatch` lze řetězit. Metoda vyvolá výjimku `std::logic_error`, pokud byl výsledek zápasu mezi `contestant1` a `contestant2` již dříve zadán.

`isOrdered (comparator)`
: metoda rozhodne, zda vyplněné výsledky zápasů vedou k jednoznačné výsledkové listině nebo ne. Návratovou hodnotou je `true` pokud je výsledková listina jednoznačná, `false` pokud ne. Parametrem volání je `comparator` - funkce, funktor nebo lambda výraz, který bude volán při získávání výsledku zápasu (převede M_ na záporné číslo / nulu / kladné číslo podle popisu výše). Metoda nesmí modifikovat instanci `CContest` a nesmí házet výjimky.

`results (comparator)`
: metoda vypočte výsledkovou listinu. Výsledkem je STL seznam (`std::list`) řetězců se jmény soupeřů v pořadí od prvního (celkového vítěze) k poslednímu. Pokud nelze výsledkovou listinu sestavit (není jednoznačná), metoda vyhodí výjimku `std::logic_error`. Parametr `comparator` má stejný význam jako u metody `isOrdered`.

Odevzdávejte zdrojový kód s implementací šablony třídy `CContest`. Za základ implementace použijte přiložený zdrojový kód. Pokud v kódu ponecháte bloky podmíněného překladu, lze takový zdrojový kód lokálně testovat a zároveň jej odevzdávat Progtestu.

Hodnocení je rozděleno na povinnou a bonusovou část. V povinné části se testují instance s malým počtem soupeřů a zápasů. Pro jeho úspěšné zvládnutí stačí základní algoritmus pracující v čase `počet_soupeřů x počet_zápasů`. Pro zvládnutí bonusového testu je potřeba použít algoritmus efektivnější.

- Pro základní řešení se může hodit vyhledání všech přímo nebo zprostředkovaně poražených soupeřů. Toto dokáže algoritmus BFS (prohledávání do šířky).
- Do parametry vyhozené výjimky zadejte nějaký popis problému (např. "duplicate match"). Fakticky se ale text ve vyhozené výjimce nekontroluje.

Vzorová data: Download

Odevzdat: Choose File No file chosen Odevzdat

Referenční řešení

Hodnotitel: automat

Program zkompilován

✓	100/100 1.00	Závazný test: **Zakladni test s parametry podle ukazky**	0.001s / 10.000s
✓	100/50 1.00	Závazný test: **Test nahodnymi daty**	0.845s / 9.999s
✓	100/100 1.30	Bonusový test: **Test rychlosti**	8.755s / 15.000s

Celkové hodnocení: 130.00 % (= 1.00 * 1.00 * 1.30)

Celkové procentní hodnocení: 130.00 %

Bonus za včasné odevzdání: 0.50

Celkem bodů: 1.30 * (5.00 + 0.50) = 7.15

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**6**	--	--	--
Řádek kódu:	**100**	16.67 ± 6.18	24	`CContest::isOrdered`
Cyklomatická složitost:	**28**	4.67 ± 2.36	7	`CContest::isOrdered`