Programozás II. ZH

SZTE Szoftverfejlesztés Tanszék

2023. ősz

Technikai ismertető

- A programot C++ nyelven kell megírni.
- A megoldást a *Bíró* fogja kiértékelni.
  - A Feladat beadása felületen a Feltöltés gomb megnyomása után ki kell várni, amíg lefut a kiértékelés. **Kiértékelés közben nem szabad az oldalt frissíteni vagy a Feltöltés gombot újból megnyomni** különben feltöltési lehetőség veszik el!
- Feltöltés után a *Bíró* a programot `g++` fordítóval és a
  `-std=c++17 -static -O2 -DTEST_BIRO=1`
  paraméterezéssel fordítja és különböző tesztesetekre futtatja.
- A program működése akkor helyes, ha a tesztesetek futása nem tart tovább 5 másodpercnél és hiba nélkül (0 hibakóddal) fejeződik be, valamint a program működése a feladatkiírásnak megfelelő.
- A *Bíró* által a `riport.txt`-ben visszaadott lehetséges hibakódok:
  - Futási hiba 6: Memória- vagy időkorlát túllépés.
  - Futási hiba 8: Lebegőpontos hiba, például nullával való osztás.
  - Futási hiba 11: Memória-hozzáférési probléma, pl. tömb-túlindexelés, null pointer használat.
- A riport.txt és a fordítási log fájlok megtekinthetőek az alábbi módon:
- A programot 20 alkalommal lehet benyújtani, a megadott határidőig.
- A programban szerepelhet `main` függvény, amely a pontszámításkor nem lesz figyelembe véve. Azonban ha fordítási hibát okozó kód van benne az egész feladatsor 0 pontos lesz.

Általános követelmények, tudnivalók

- Csak a leírásban szereplő osztályokat, metódusokat és adattagokat kell megvalósítani, egyéb dolgokért nem jár plusz pont.
- Minden metódus, amelyik nem változtatja meg az objektumot, legyen konstans! Ha a paramétert nem változtatja a metódus, akkor a paraméter legyen konstans!
- string összehasonlításoknál az egyezés a pontos egyezést jelenti, azaz ha kis-nagy betűben térnek el, akkor már nem tekinthetők egyenlőnek (pl. a "piros" != "Piros")
- A leírásokban bemutat példákban a string-ek köré rakott idézőjelek nem részei az elvárt kimenetnek, azok csak a string határait jelölik. Például ha az szerepel, hogy a példa bemenetre az elvárt kimenet az, hogy `"3 alma"`, akkor az elvárt kimenet idézőjelek nélkül az `3 alma`, de a szóköz szükséges!
  - A tesztesetekben nem lesz ékezetes szöveg kiíratása.
- Az 1. és 2. ZH során STL tárolók használata nem megengedett! Tárolók használata 0 pontot eredményez.
- Az elvárt kimeneteknek karakterről karakterre olyan formátumúnak kell lennie, ami a feladatban le van írva (szóközöket és sortöréseket is beleértve).
- Ha az objektum másolása nem triviális (azaz a fordító által generált másolás nem elegendő), akkor a megfelelő másolást is meg kell valósítani.

Kiindulási projekt, megoldás feltöltése

- **A megoldáshoz az előre kiadott osztályok módosítása szükséges lehet.**
  - Nem minden ZH esetében van kiindulási projekt.
- Feltöltéskor ezeket az osztályokat is fel kell tölteni és a módosításokat is pontozhatja a bíró!
- Egyes tesztesetekben a bíró módosított osztályt is használhat ezen kiinduló osztályok helyett, ezzel tesztelve a valóban helyes működést!

Zh alatt használható segédanyag

- **A ZH során használható segédanyag elérhető Biro-ban.**

A cppreference off-line változata a biroban elérhető a következő linken: `https://biro.inf.u-szeged.hu/kozos/prog2/html-book-20220730.zip`.

Vector increment (5 pont)

Készíts egy `vector_increment` nevű függvényt! A függvény be-/kimenő paramétere legyen egy inteket tároló vector! (Be-/kimenő paraméter ami az eredeti objektumon dolgozik és azt módosítani is tudja.) A függvény második paramétere egy int; ez egy limit lesz.

A függvény a vector minden elemét, ami nagyobb mint a limit (függvény 2. paramétere) növelje meg az eredeti vector átlagával.

Count_if_wrapper (5 pont)

Valósítsd meg a count_if_wrapper metódus törzsét! **A count if wrapper szigorúan csak a `std::count_if` metódust használhatja!**

A metódus adja vissza azon elemek darabszámát, melyek kétszerese nagyobb mint a függvény második paramétere!

Set (2 + 5 pont)

Egészítsd ki a Label osztályt, hogy azt használni lehessen a set adatszerkezetben! A rendezés alapja a Label szövegének hossza. **(1+1 pont)**

Készíts egy `descending_set_print` nevű függvényt! A függvény egy Label-eket tároló set-et várjon paraméterül! A függvény a standard kimenetre írja ki a set-ben tárolt label-ek szövegét! Minden label szövege után új sor legyen! **A Label-ek ne az alap növekvő, hanem csökkenő hosszúság szerint legyenek kiírva!**

Sort partial (6 pont)

Valósítsd meg a `sort_partial` függvény törzsét! A függvény rendezze sorba a vector azon elemeit melyek a search elem után vannak, a search elemet nem belevéve! A rendezésben résztvevő intervallum: (<pos>, end) ahol <pos> a search elem.

Degree map (10 pont)

Valósítsd meg a `calculate_degree` függvény törzsét!

A paraméterben kapott map egy gráf hívási listája. A kulcs unsigned érték a hívó csúcspont azonosítója, a hozzátartozó unsigned érték a hívott csúcspont azonosítója. A függvény adjon vissza egy map-et ahol a kulcsok az előbb említett gráf csúcspontjainak azonosítói (az összes) és az értékek az adott csúcs bemenő fokszáma.

Bemenő fokszám: hány hívás végpontja az adott csúcs.

Minden csúcspont legyen benne az eredmény map-ban, akkor is ha csak 0 a bemenő fokszáma!

A gráfban ha egy csúcshoz nincsen kimenő él, akkor a hívott csúcs azonosítója 0. A 0s azonosítójú csúcs nem kell, hogy szerepeljen az eredmény map-ben! (Ne legyen ilyen kulcs!)