Adatszerkezetek és algoritmusok 2.
PÓT ZH
1. FELADAT

Tájékoztató

Kérjük, hogy ezt a tájékoztató szöveget figyelmesen elolvasni.
A ZH maximális ideje négy óra (240 perc).
A ZH két részből áll:

1. Az első feladat megoldása során kizárólag a papír, írószerszám, PC (Eclipse IDE) használható. A feladat megoldása során a lehetséges 40 pontból legalább 30 pontot kell elérni. (A részpontszámok a feladatban meg vannak adva.) A feladat értékelése a zh közben történik meg, aminek a következő lépései vannak: Szólni kell a felügyelőnek, aki átnézi és meghatározza, hogy hány pontot ér a megoldás. Amennyiben a javításra szorul még lehet dolgozni rajta, ám mindenki legfeljebb háromszor kérhet értékelést. Ha a 40 pontból legalább 30 pontot sikerült megszerezni, akkor a zh folytatható a második feladattal, azonban a zh folytatása esetén az első feladat pontszáma véglegessé válik!
2. A második feladat során az előzőeken túl a kiadott zip fájlban található segédanyagokat lehet használni (Előadás diák, gyakorlat diák, gyakorlat kódok.) A második feladat megoldását a ZH-t követően értékeljük, előzetesen becslést az eredményekre nem adunk.

A ZH összesen 100 pont, amiből az elégséges szinthez 50 pontot kell elérni, ezen belül az első feladatból legalább 30 pontot kell elérni.

A kiadott segédanyagokat a csatolt hálózati meghajtón lehet megtalálni található.
M**eg nem engedett segédeszköz használata az aláírás megtagadását vonja maga után.**

A feladatokat figyelmesen olvasd végig, tervezd meg a kódot és csak ezután készítsd el. A programnak futnia kell és a kódnak a kivételeket kezelnie kell.

Mindkét feladat megoldását be kell adni!

- A beadott fájlokat NE tömörítsétek.
- A projekt neve az egyetemi hálózati azonosítóval kezdődjön!
- A megoldások egyértelműen megkülönböztethetők legyenek!
- Csak a projektet adja be mindenki!

1. feladat (40 pont összesen)

Írj egy olyan kupacot, amiben a legkisebb elem található a kupac tetején. A feladat megoldása során az std::priority_queue és std::make_heap nem használható. (Azonban dinamikus konténer igen (std::vector).)

A feladat megoldása során ügyelj a következőkre:

- A kupac adatszerkezet (publikus) függvényeit implementálni kell!
- A kupacban a MINIMÁLIS elem van a kupac tetején!
- Csak a szükséges algoritmusokat és funkciókat kell implementálni!

A kész kupacot a programban teszteld. (A lehetséges függvények mindegyikét, több különböző inputra.)

Az előzőekben készített kupacot használd rendezésre.

A feladat pontozása:
A kupac reprezentáció helyessége: 5 pont
A kupac tetejének helyes megválasztása: 5 pont
A kupacban elem elhelyezése: 10 pont
A kupacból elem kivétele: 10 pont
A kupac tesztelése: 5 pont
A kupaccal történő rendezés: 5 pont

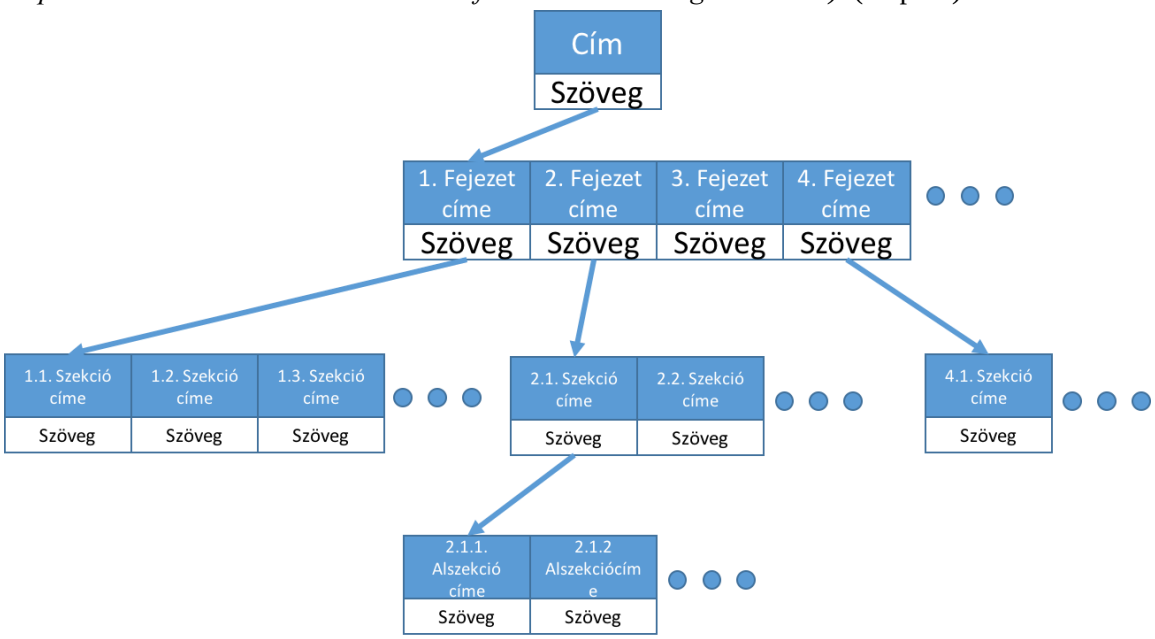
Adatszerkezetek és algoritmusok 2.
PÓT ZH

2. FELADAT

2. feladat (60 pont összesen)

FONTOS, hogy a feladatot olvasd végig és értelmezd!

a) Valósíts meg egy fa adatszerkezetet, amelyben minden csúcsnak tetszőleges számú gyereke lehet. Adott csúcs gyerekei maradjanak a beszúrás sorrendjében. Minden csúcsban két szöveges adat mező legyen, cím és szöveg. Ezáltal lehetővé válik egy dokumentum (pl. könyv) tárolása, illusztrációnak lásd az ábrát. *(Figyelem! A beszúrás algoritmusa a b) pontban van leírva, ebbe a tíz pontba az nem számít bele. Törlést a fában nem kell megvalósítani.)* (10 pont)



b) A fának legyen egy beszúrás művelete, mely adott csúcshoz be tud egy új gyereket szúrni. Olvass be egy könyvet ebbe a fába a “text.txt” fájlból, ehhez használhatod a mellékelt kódot (reader.cpp). A könyvnek van egy címe, és fejezetekből (Chapter) szekciókból (Section) és alszekciókból (Subsection) áll, ezek általános neve a tartalmi egység, ezek adják a fa szintjeit: a gyökérelem tárolja a címet, annak gyerekei a fejezetek, azoknak pedig a szekciók, stb. Az egyes tartalmi egységeknek van címük illetve lehet nekik szövegük (de nem szükségszerűen), ezek felelnek meg a csúcsokban tárolt adatmezőknek. Köztes szint nincs kihagyva, tehát az nem lehet hogy egy Chapteren belül rögtön Subsection jön, de a fa nem mindenhol teljes mélységű (tehát lehet pl. Section amiben nincs Subsection).Az egyes tartalmi egységek címe a test szövegétől sortöréssel van elválasztva, a következő alakban: '!Title: <cim>', '!Chapter: <cim>' stb. alaku, ahol a '<cim>' az adott egység címe, a megadott kód (reader.cpp) ezeket találja meg a bemeneti fájlban - lásd az alábbi példaszöveget. (25 pont)

Segítség a beszúráshoz: mivel a könyv alapból rendezett, a beszúrás során a fa egy adott pontján állunk (amit épp beszúrtunk), ami lehet 0 - cím, 1 - fejezet, 2 - szekció, 3 - alszekció. A beszúrandó

elem szintje vagy eggyel nagyobb, ekkor az aktuális elem (első) gyerekének szúrjuk be, vagy megegyező szintű ekkor az aktuális elem következő testvére lesz (azaz a szülő következő gyereke), vagy pedig alacsonyabb szintű, ekkor pedig a különbségnek megfelelő szintet kell fel lépni és beszúrni mint következő gyereket. A beszúrást tehát ilyen "memóriával" érdemes megoldani.

c) A fának legyen egy metódusa mellyel ad adott csúcshoz le tudja kérdezni, hogy hány karaktert tartalmaz az adott tartalmi egység, beleértve a címek hosszát is, de figyelmen kívül hagyva a sortöréseket. (10 pont)

d) Készíts el egy metódust, amely elkészíti a könyv tartalomjegyzékét, és kiírja a konzolra, hierarchikus formában, hierarchikusan számozott tartalmi egységekkel (tehát az első fejezet második szekciójának harmadik alszekciója: 1.2.3. <címe>), valamint címükkel és oldalszámukkal. Az oldalszám kiszámítása során az előző összes egység karaktereinek számával történik, egy oldalra pontosan 500 karakter fér ki (címeket és szöveget is beleértve, a sortöréseket viszont nem). (15 pont)

Példa szöveg

!Title: Lorem Ipsum

!Chapter: Dolor Sit

Amet, consectetur adipiscing elit. Praesent felis nisl, mattis at velit.
Maecenas risus enim, malesuada ac tempus eget, iaculis varius nisi.

!Section: Cras pharetra

!Subsection: Quam quis gravida varius

Ligula nisl varius nulla, eget gravida mauris ex in ante. Integer consequat auctor purus, at consequat neque tempus eget. Vestibulum eu aliquam sem, vehicula tincidunt magna. Mauris laoreet placerat dictum. Aenean et molestie orci.

!Subsection: Praesent rutrum

Quam leo, vel lobortis mi euismod sed. Maecenas sagittis sapien mauris, sed mollis tortor euismod quis. Pellentesque nec est nisl.

!Chapter: Proin sed tincidunt tellus.

!Section: Suspendisse scelerisque

Mollis lorem sit amet fermentum.

!Section: Nunc bibendum

!Subsection: Lacus sed

Turpis dictum, et tempor nisi dignissim. Aenean consequat erat eleifend tincidunt lacinia. Praesent lacinia tempus nisl id pulvinar. Cras commodo est vitae viverra viverra.