09.71.23/24 Tölvunarfræði 2/2a

Lokapróf **11. maí, 2001**
Kennari: Hjálmtýr Hafsteinsson **kl. 13^{30} ? 16^{30}**

Fyrstu 5 dæmin eru fyrir alla nemendur (bæði í Tölvunarfræði 2 og 2a) Dæmi 6 er aðeins fyrir nemendur í Tölvunarfræði 2, en dæmi 7 er aðeins fyrir nemendur í Tölvunarfræði 2a (verkfræði-nema). Í báðum tilfellum gilda ***fimm bestu dæmin af sex****. Dæmin hafa jafnt vægi.*

Öll skrifleg hjálpargögn og reiknivél leyfð*.*

?? Athugið að þegar beðið er um að "Lýsa" eða "Sýna" þá er nóg að gera það í orðum og með teikningum. Ef þið eigið að skrifa C++ kóda þá er beðið um það sérstaklega.
?? Rökstyðjið öll svör og munið að það er óþarfi að skrifa upp skilgreiningar sem eru í bókinni.

1. Í *vegþjöppun með helmingun* (path compression by halving, Forrit 1.4 í bók) fyrir Union-Find verkefnið er aðeins **annar hver hnútur** á leiðinni frá `p` (og `q`) uppí rót látinn benda á afa sinn (þ.e. foreldri foreldri síns).

a) Sýnið að þetta er það sem gerist, með 6 hnúta dæmi, og útskýrið hvers vegna Forrit 1.4 hoppar svona yfir annan hvern hnút.

b) Útfærið (í C++) útgáfu þar sem **allir** hnútar á leiðinni frá `p` uppí rót eru látnir benda á afa sinn.

2. Gefinn er raðaður hringtengdur (eintengdur) listi *L* með haushnút, þar sem aftasti hnútur listans bendir á haushnútinn. Skrifið fall í C++, sem tekur inn bendi á *L* (þ.e. á haushnút hans) og gildi *x*, og skiptir *L* upp í tvo hringtengda lista *A* og *B*, þannig að *A* inniheldur þá hnúta sem hafa gildi ? *x*, en *B* þá hnút með gildi > *x*. Athugið að hnútar *L* eru fluttir yfir í *A* eða *B*. Hér að neðan er haus fallsins.

```
void skipta( node* L, int x, node* &A, node* &B )
```

3. Í venjulegri *inorder*-röð er alltaf farið fyrst niður vinstra barn, síðan í hnútinn sjálfann, og loks niður í hægra barn. Við getum hugsað okkur aðra röð á þessu, sem við köllum *outorder*-röð, þar sem fyrst er farið niður í **hægra** barn, síðan í hnútinn sjálfann, og loks niður í **vinstra** barn.

Er nægilegt að vita inorder- og outorder-röð hnúta tvíundartrés til að ákvarða það einkvæmt? Með öðrum orðum: eru til fleiri en eitt tvíundartré sem eru bæði með sömu indorder-röð og sömu outorder-röð? Lýsið aðferð til að finna tréð eða sýnið mótdæmi.

4. Hægt er að setja venjuleg tvíundartré upp í vektor á sama hátt og hrúgur.

a) Nefnið þær breytingar á framsetningu sem þarf að gera vegna þess að venjuleg tvíundartré eru ekki eins regluleg og hrúgur.

b) Hver er hámarksminnisnotkun fyrir *N*-hnúta tvíundartré í versta tilfelli?

c) Sýnið C++ fall sem prentar út *postorder* röð hnúta í tvíundartré með þessari uppsetningu.

5. Ef við leyfum innsetningu margra stak með sama gildið inn í *tvíleitartré* þá getur verið gott að vita hversu mörg stök hafa tiltekið gildi í trénu.

a) Útskýrið nákvæmlega hvar stök með sama gildi lenda í tvíleitartré (þ.e. miðað við hvert annað). Gróf teikning ásamt útskýringu ætti að nægja.

b) Lýsið nákvæmlega útfærslu á fallinu `count( v )`, sem skilar fjölda staka í trénu með gildið `v`. Aðferðin verður að vera hraðvirkari en að fara bara í gegnum allt tréð og telja stök með gildið `v`. (Vísbending: Notið aðferð sem er í tveimur þrepum)

Aðeins fyrir nemendur í Tölvunarfræði II:

6. Í Shell-röðun er notuð Innsetningaröðun til að grófraða vektornum, þ.e. *h*-raða vektornum fyrir sífellt lækkandi *h*, þar til *h*=1.

a) Væri hægt að nota Bóluröðun (Bubble sort) í stað Innsetningaröðunar? Lýsið í orðum og teikningum hvernig sú útfærsla af Shell-röðun væri og í hverju hún væri frábrugðin upphaflegu Shell-röðunaraðferðinni.

b) Hvað með Quicksort í stað Innsetningaröðunnar í Shell-röðun? Lýsið göllum þeirrar útgáfu og hvaða vandkvæði væru við útfærslu. Er líklegt að þessi aðferð væri hraðvirkari en upphaflega Shell-röðunin?

Aðeins fyrir nemendur í Tölvunarfræði IIa (verkfræðinema):

7. Hægt er að skilgreina nýja gerð hrúga (heap), svokallaða *tvívíða hrúgu*, fyrir stök af taginu (i, j) á eftirfarandi hátt: Fyrir hnút *v* á **jöfnu dýpi** er gildi **fyrra hnits** staksins stærra en gildi fyrra hnits í öllum hnútum í hluttrénu undir *v*. Á sama hátt gildi að fyrir hnút *v* á **oddatölu dýpi** er gildi **seinna hnits** staksins stærra en gildi seinna hnits í öllum hnútum í hluttrénu undir *v*. Það eru síðan tvær útgáfur af fallinu `getmax` eftir því hvort við viljum stakið með stærsta fyrra hnit eða stærsta seinna hnit.

a) Sýnið mögulega tvívíða hrúgu sem inniheldur stökin (3, 8), (7, 1), (10, 5), (9, 2), (5, 7).

b) Lýsið útgáfunni af `getmax` fyrir seinna hnit.

c) Lýsið innsetningu staks í tvívíða hrúgu.