Kirjoja, laskinta tms. ei saa olla tentissä. Tehtävällä on yksi tai useampi numero maksimipistemäärän mukaan. Muiden kuin koodaa tai piirrä -tehtävien mallivastaukset ovat melko lyhyet, tyypillisesti 1, ..., 3 riviä pistettä kohti.

1. Puolitushausta on olemassa eri versioita. Miksi joitakin niistä ei voida testata siten, että etsitään avaimen paikka taulukosta jollain muulla, luotettavalla tavalla, ja verrataan testikohteen vastausta siihen?

   Avain saattaa olla taulukossa useasti, ja testikohde ja luotettava etsintä saattavat löytää eri kohdat.

2&3. Luettele kolme virhettä tai muuta huonoutta oheisessa yrityksessä toteuttaa eräs algoritmi. Kerro jokaisesta virheestä tai huonoudesta, mitä haittaa se aiheuttaa, olettaen että muut virheet on korjattu mutta sitä ei.

```
1  for i := 0 to A.koko − 1 do
2    apu := A[i]; j := i
3    while A[j − 1].x ≥ apu.x && j > 0 do
4      A[j] := A[j − 1]; j = j − 1
5    A[j] := apu
```

   1. Osat $A[j-1].x \geq apu.x$ ja $j > 0$ ovat väärässä järjestyksessä. Kun $j = 0$, saatetaan indeksoida $A[-1]$ ja siten kaataa ohjelma.
   2. Kierros $i = 0$ ei tee mitään hyödyllistä, joten se kuluttaa turhaan aikaa.
   3. Vertailu "$\geq$" pitäisi olla "$>$". Koska se ei ole, vakaus menetetään. Jos on yhtäsuuria alkioita, niin myös nopeutta menetetään.

4. Mikä on heapsortin (kekojärjestäminen) tärkein etu muihin nopeisiin järjestämisalgoritmeihn verrattuna?

   Pieni lisämuistin tarve.

5. Mikä on mergesortin (lomitusjärjestäminen) tärkein etu muihin nopeisiin järjestämisalgoritmeihn verrattuna?

   Vakaus.

Tentissä kysyttiin erään algoritmin ajan kulutusta. Joku vastasi $O(n\sqrt{n})$ ja joku toinen $\Theta(n\sqrt{n})$.

6. Onko mahdollista, että $O(n\sqrt{n})$ on matemaattisesti oikein mutta $\Theta(n\sqrt{n})$ matemaattisesti väärin? Perustele vastauksesi lyhyesti.

   Näin on esimerkiksi jos todellinen suoritusaika on $\Theta(n)$.

7. Onko mahdollista, että $\Theta(n\sqrt{n})$ on matemaattisesti oikein mutta $O(n\sqrt{n})$ matemaattisesti väärin? Perustele vastauksesi lyhyesti.

   Ei ole. Suoritusaika on $\Theta(n\sqrt{n})$ jos ja vain jos se on sekä $O(n\sqrt{n})$ että $\Omega(n\sqrt{n})$.

8. Voiko jompikumpi vastaus olla matemaattisesti oikein mutta ei silti ansaitse ainakaan täysiä pisteitä? Perustele vastauksesi lyhyesti.

   Kyllä. Jos suoritusaika on $\Theta(n)$, niin $O(n\sqrt{n})$ on matemaattisesti oikein, mutta ei tarkin mahdollinen.

9. Kirjoita pseudo- tai ohjelmakoodina yksinkertainen algoritmi, joka havainnollistaa tehtävän 6, 7 tai 8 vastaustasi. Kerro, minkä tehtävän vastausta se havainnollistaa.

```
1  x := 0
2  for i := 1 to n do x := x + i
```

   Havainnollistaa tehtävää 8.

Bellman–Ford-algoritmi käsittelee suunnattuja graafeja, joiden kullakin kaarella on pituus. Sen yksinkertaisin versio tekee erään määrän kierroksia. Kullakin kierroksella se käy kaikki kaaret läpi. Jos tutkittavan kaaren kautta saadaan aikaisemmin löydettyjä lyhyempi reitti kaaren loppupään solmuun, niin se merkitään solmun tietoihin. Jos mahdollisimman lyhyt reitti lähdöstä maaliin on olemassa, niin se halutaan lopuksi tulostaa.

10. Mitkä ovat kaksi tärkeintä suunnatun graafia kokoa koskevaa tietoa? Vastaukseksi ei riitä antaa kaksi muuttujan nimeä, vaan pitää myös kertoa sanallisesti, mitä ne tarkoittavat.

    $|V|$ on solmujen määrä ja $|E|$ on kaarten määrä.

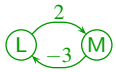
11. Ilmaise kierrosten määrä syötteenä annetun graafin kokoa koskevien tietojen avulla.

    $|V|$

12. Ilmaise kullakin kierroksella tehtävä työmäärä O-, Θ- tms. merkinnällä.

    $\Theta(|E|)$

13. Piirrä esimerkki suunnatusta graafista, jossa on reitti lähtösolmusta maalisolmuun mutta ei ole mahdollisimman lyhyttä reittiä.



14. Mitä tietoja pitää liittää kuhunkin solmuun Bellman–Ford-algoritmin toteutuksessa?

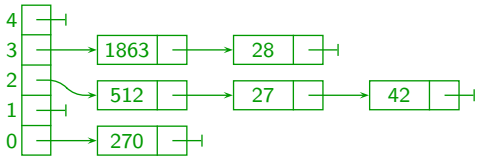
    Lyhin löydetty etäisyys solmuun sekä linkki edelliseen solmuun ko. reitillä.

15. Mitä tietoja pitää liittää kuhunkin kaareen Bellman–Ford-algoritmin toteutuksessa?

    Pituus; sen solmun numero, josta kaari alkaa; sen solmun numero, johon kaari loppuu.

Seuraavissa tehtävissä tarkastellaan hajautustauluja.

16. Piirrä hajautustaulu, jossa hajautusfunktiona on $n \bmod 5$ ja tallennettuna ovat 27, 28, 42, 270, 512 ja 1863.



17. Minkälaisessa tilanteessa etsintä avaimen perusteella hidastuu olennaisesti?

    Kun hajautuslistoista tulee pitkiä.

18. Minkälaisessa tilanteessa muistia kuluu paljon suhteessa hyötytiedon määrään?

    Kun hajautuslistoja on paljon enemmän kuin tallennettuja alkioita.

19. Minkä palvelun tasapainotettu binäärihakupuu tuottaa nopeasti, mutta hajautustaulu ei tuota?

    Selaaminen avainten suuruusjärjestyksessä.

Seuraavaksi käsitellään binäärihakupuita. Ota huomioon, että binäärihakupuu saa olla tyhjä!

20. Kirjoita ei-rekursiivinen algoritmi, joka saa osoittimen s juureen ja etsii suuruusjärjestyksessä viimeisen solmun.

```
if s then
   while s->oikea do s := s->oikea
return s
```

21. Kirjoita rekursiivinen algoritmi, joka saa osoittimen juureen ja palauttaa puun korkeuden.

```
korkeus(s)
if s = nil then return −1
v := korkeus(s->vasen)
o := korkeus(s->oikea)
if v > o then return v + 1
return o + 1
```

22. Kerro sanallisesti, miten solmusta alkaen voidaan tehokkaasti löytää suuruusjärjestyksessä seuraava solmu.

Jos puu on tyhjä, palautetaan nil. Jos solmulla on oikea lapsi, niin mennään sinne ja jatketaan alas vasempaan lapseen, sen lapseen ja niin edelleen kunnes tullaan solmuun, jolla ei ole vasenta lasta. Muussa tapauksessa mennään solmun vanhempaan, sen vanhempaan ja niin edelleen kunnes tullaan johonkin solmuun sen vasemmasta lapsesta tai tullaan ulos puusta (juuren olemattomaan vanhempaan).

23&24. Kirjoita ei-rekursiivinen algoritmi, joka saa osoittimen s solmuun ja etsii suuruusjärjestyksessä seuraavan solmun.

```
if s = nil then return nil
if s->oikea then
   s := s->oikea
   while s->vasen do s := s->vasen
   return s
e := s;  s := s->vanh
while s ja e = s->oikea do
   e := s;  s := s->vanh
return s
```

Pudotuskokeita varten on varattu n testikappaletta. Pudotuksessa testikappale joko särkyy tai säilyy täysin ehjänä. Testikappaletta voi käyttää uudessa pudotuksessa jos ja vain jos se ei särkynyt. Jos jokin testikappale särkyy pudotettaessa joltakin korkeudelta, niin jokainen testikappale särkyy pudotettaessa siltä tai suuremmalta korkeudelta. Eri korkeuksia on k kappaletta.

Oheinen aliohjelma suunnittelee koesarjan, jolla saadaan mahdollisimman vähillä pudotuksilla selville pienin korkeus, jolta pudotettaessa testikappaleet särkyvät, tai tieto, että testikappaleet kestävät pudotuksen suurimmaltakin korkeudelta. Aliohjelma palauttaa enimmillään tarvittavan pudotusten määrän. Alkuperäinen kutsu on `luo_taulukko valinta[k+1][n+1]; int t = tulos( k, n );`. Jos $k > 0$ ja $n > 0$, niin aliohjelman lopetettua `valinta[k][n]` kertoo, miltä korkeudelta pitää pudottaa, siten että 0 on matalin niistä korkeuksista jotka vielä pitää tutkia, 1 on toiseksi matalin ja niin edelleen. Rivillä 3 palautettava `ääretön` kertoo, että tehtävä on mahdoton, koska 0 testikappaletta ei riitä tarpeeksi moneen pudotukseen.

```
1  int tulos( int k, int n ){
2    if( k == 0 ){ return 0; }
3    if( n == 0 ){ return ääretön; }
4    int min = ääretön;
5    for( int i = 0; i < k; ++i ){
6      int uusi = maksimi( tulos( i, n-1 ), tulos( k-i-1, n ) );
7      if( uusi < min ){ min = uusi; valinta[k][n] = i; }
8    }
9    return min+1;
10 }
```

25. Meneekö `tulos` rikki, jos rivit 2 ja 3 vaihdetaan keskenään? Perustele.

Menee. Nollan korkeuden tutkiminen nollalla testikappaleella onnistuu ja tarvitsee nolla pudotusta. Jos rivit vaihdettaisiin, `tulos` vastaisi virheellisesti ”ei onnistu”.

26&27. Miksi rivin 6 rekursiivisten kutsujen parametreina on i ja $n-1$ sekä $k-i-1$ ja n?

Jos testikappale särkyi, niin jäljellä on $n-1$ testikappaletta, ja on tutkittava ne i korkeutta, jotka ovat pienempiä kuin rivin 6 pudotuksessa käytetty. Jos testikappale ei särkynyt, niin jäljellä on n testikappaletta, ja on tutkittava ne $k-i-1$ korkeutta, jotka ovat suurempia kuin rivin 6 pudotuksessa käytetty.

28. Miksi `tulos` on erittäin hidas jo melko pienillä k ja n?

Se laskee samoja välituloksia uudelleen ja uudelleen.

29&30. Luonnostele muutokset, joilla `tulos` saadaan paljon nopeammaksi. Kerro mitä ja mihin kohtiin (esim. rivien 1 ja 2 väliin) lisätään, muutetaan ja/tai poistetaan.

Lisätään globaali taulukko `pudotuksia` kooltaan $k+1, n+1$. Rivien 3 ja 4 väliin lisätään `if( !pudotuksia[k][n] ){`. Rivi `return min+1;` muutetaan muotoon `pudotuksia[k][n] = min+1; } return pudotuksia[k][n];`.

Loppu