L11 Tehtävät

- Algoritmi, suorituskyky, ongelman ratkaisu, suunnittelu, tiedon etsiminen

Lue oppaan tämän viikon asioita käsittelevä luku 11. Lisäksi tehtävien suorittamiseen tarvitset aiempien lukujen tietoja. Ohjelmointitehtävissä on oltava otsikkotiedot ja ne palautetaan Moodlen kautta CodeGradeen.

L11T1: Algoritmikehitys: Algoritmi neliöjuuren laskentaan

Ensimmäisessä tehtävä on kehittää algoritmi ja ohjelma luvun neliöjuuren laskentaan. Kysy ensin käyttäjältä luku. Tämän jälkeen tee silmukka, jonka avulla etsit kokonaisluvun, joka toiseen korotettuna on lähimpänä käyttäjän antamaa lukua. Koska vastaus annetaan kokonaislukuna, voi tulostuksessa olla pientä heittoa oikeaan arvoon nähden.

Ohjelman esimerkkiajo 1:

```
Anna luku: 144
Neliöjuuri on 12
Kiitos ohjelman käytöstä.
```

Ohjelman esimerkkiajo 2:

```
Anna luku: 115
Neliöjuuri on 11
Kiitos ohjelman käytöstä.
```

L11T2: Mallin ohjelmointi: Kanipopulaatio ja Fibonaccin sarja

Selvitä kanipopulaation kasvua optimitilanteessa simuloimalla. Vuoden alussa autiolla, mutta rehevällä, saarella on yksi vastasyntynyt kaniparisku nta. Kanit pystyvät lisääntymään kuukauden vanhana ja raskaus kestää yhden kuukauden. Jokainen synnytys tuottaa uuden kaniparisku nnan. Tehtävänäsi on luoda ohjelma, jolla voidaan selvittää, kuinka monta kaniparisku ntaa saarella on x kuukauden kuluttua. Kuten ehkä jo huomasitkin, kanien lisääntyminen tällaisessa ideaalimaailmassa noudattaa Fibonaccin sarjaa (http://fi.wikipedia.org/wiki/Fibonaccin_lukujono).

Tämä tehtävä on tyypillinen esimerkki reaalimaailman simuloinnista algoritmin avulla. Simulointi lähtee liikkeelle yksinkertaisesta matemaattisesta mallista, jonka oletetaan toimivan ideaalitilanteessa. Usein malli voi olla alussa kaukana todellisuudesta ja useita uusia tekijöitä on huomioitava mallissa ennen kuin se on oikeasti lähellä todellisuutta. Tyypillisesti mallin laajennus perustuu laajoihin empiirisiin kokeisiin ja niiden vaikutusten huomiointiin laskennassa, eivätkä ne ole tämän ohjelmointikurssin asioita. Tällä kurssilla kokeillaan vain yksinkertaisen matemaattisen mallin ohjelmointia.

Ohjelman esimerkkiajo 1:

```
Anna kuukausien lukumäärä: 4
Kaniparisku ntia on 4 kuukauden kuluttua 5
```

```
Kiitos ohjelman käytöstä.
```

Ohjelman esimerkkiajo 2:

```
Anna kuukausien lukumäärä: 9
Kanipariskuntia on 9 kuukauden kuluttua 55
Kiitos ohjelman käytöstä.
```

L11T3: Rekursiivinen algoritmi

Tee ohjelma, joka kysyy käyttäjältä tulostettavan sanan ja tulostuskertojen lukumäärän ja tulostaa sitten annetun sanan esimerkkiajon mukaisesti annetun lukumäärä-kertaa rekursiiviseen ratkaisuun perustuen. Ohjelmaan kannattaa laittaa pääohjelma, joka kutsuu aliohjelmaa, joka hoitaa tulostuksen rekursion avulla. Rekursiivinen aliohjelma on varsin lyhyt ja rekursion ytimenä kannattaa käyttää em. lukumäärää. Huomaa, että rekursiivisen kutsun ja tulostamisen järjestyksellä on väliä. Tämä ohjelma on siis lyhyt ja yksinkertainen, mutta edellyttää rekursio-idean ymmärtämistä ja toteuttamista.

Huomaa, että toistorakenteeseen tms. perustuva ratkaisu ei käy tässä tehtävässä vaan ratkaisun on perustuttava rekursiiviseen aliohjelmaan. Tutustu asiaan tarvittaessa luentojen ja ohjelmointioppaan avulla.

Ohjelman esimerkkiajo:

```
Anna tulostettava sana: moi
Anna tulostuskertojen määrä: 3
Sana on 'moi', 1. kerta.
Sana on 'moi', 2. kerta.
Sana on 'moi', 3. kerta.
Kiitos ohjelman käytöstä.
```

L11T4: Nopea hakualgoritmi

Tehtävänäsi on tehdä annettuun ohjelmatiedostoon hakufunktio, joka testaa pitkästä lukujoukosta (yli 10000 lukua), onko siinä sellaista lukuparia, jossa ensimmäinen luku on vähemmän kuin yksi kolmasosa toisesta luvusta.

- Esimerkiksi lukujonossa "10 15 26 12 22 31" tällainen pari on 10 ja 31, koska (3*10) < 31 tai toisin sanottuna 10 < (31 / 3).
- Samoin lukujonossa "15 34 49 22 27 14 40 33" tällainen lukupari olisi 15 ja 49. Myös pari 14 ja 49 kelpaisi, mikäli se löytyisi ensin.
- Lukujonosta "10 16 20 29 11 17" taas lukuparia ei löydy, koska 3*10 on suurempi kuin 29.
- Mikäli sopiva pari löytyy, voidaan aliohjelman ajo lopettaa välittömästi ja palauttaa luvut listassa pääohjelmaan.
- Jos paria ei löydy, palautetaan oletusarvot 0 ja 0 pääohjelmaan.

Tällä kertaa haemme ratkaisua, joka on mahdollisimman nopea. Siksi ratkaisulta vaaditaan, että se antaa tiedostoissa L11T4D1.txt ja L11T4D2.txt olevista lukujonoista oikean vastauksen (löytyy + numerot / ei löydy) alle 2 sekunnissa. Numerot ovat datajoukossa satunnaisessa järjestyksessä, jolloin parin suurempi luku voi myös olla pienemmän luvun edellä. Kun hakualgoritmisi löytää vastauksen, **tallenna pienempi luku listan `Luvut` ensimmäiseksi alkioksi ja suurempi toiseksi**. Tehtävä liittyy ohjelmien suoritustehokkuuteen eikä sille ole olemassa yhtä oikeaa ratkaisua.

Moodlessa L11 tehtävien tiedostoissa on Python ohjelman runko L11T4_runko.py, ota tämä tiedosto lähtökohdaksi ja lisää aliohjelmaan `hakufunktio` tarvittava koodi tiedoston lukemiseen ja lukuparin etsimiseen.

Ohjelman esimerkkiajo 1:

```
Anna tiedoston nimi: L11T4D1.txt
Hakualgoritmi oli riittävän nopea!
Se löysi sopivan parin: 30 ja 91
Kiitos ohjelman käytöstä.
```

Ohjelman esimerkkiajo 2:

```
Anna tiedoston nimi: L11T4D2.txt
Hakualgoritmi ei löytänyt sopivaa lukuparia.
Kiitos ohjelman käytöstä.
```

Ohjelman esimerkkiajo 3:

```
Anna tiedoston nimi: eiole.txt
Tiedoston 'eiole.txt' käsittelyssä virhe, lopetetaan.
```