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.

Sisällysluettelo

L11T1: Algoritmikehitys: Neperin luvun likiarvo (P)

Tehtävän taso: Perustaso

Neperin luku on matemaattinen vakio, joka on esimerkiksi luonnollisen logaritmifunktion kantaluku. Neperin luvun kaksikymmentä ensimmäistä desimaalia ovat:

$$2.71828\ \ 18284\ \ 59045\ \ 23536$$

Neperin luvulle on löydetty sarjakehitelmä, jolla sen desimaaleja voidaan tehokkaasti laskea:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

Tätä kaavaa ei kannata pelästyä, sillä se voidaan esittää ymmärrettävämmässä muodossa näin:

$$e = \tfrac{1}{0!} + \tfrac{1}{1!} + \tfrac{1}{2!} + \tfrac{1}{3!} + \ldots$$

ja vielä yksinkertaisemmin:

$$e = 1 + 1 + \tfrac{1}{1\cdot 2} + \tfrac{1}{1\cdot 2\cdot 3} + \ldots$$

Tee Python ohjelma, joka laskee Neperin luvulle likiarvon käyttämällä yllä olevaa kaavaa. Kysy käyttäjältä montako kierrosta (eli kertomaa) ohjelma laskee. Ohjelmassasi voit käyttää esimerkiksi `math` kirjastosta löytyvää `factorial()` funktiota kertoman (eli jakajan) laskemiseen. Tulosta tulos kahdenkymmenen desimaalin tarkkuudella, eli esimerkiksi formaattia '`:.20g`' (kts. ohjelmointioppaan luvusta 6 miten tulostetta voi muotoilla).

Huomaa, että kierroksien määrä ei vastaa oikeiden desimaalien määrää. Huomaa myös, että tällä tavalla laskettaessa saamme maksimissaan noin viisitoista oikeaa desimaalia, johtuen tiedon esittämisestä tietokoneen sisällä binäärilukuina. Tämä ei haittaa tässä tehtävässä, eikä asiaan paneuduta sen tarkemmin tällä kurssilla.

Esimerkkiajo 1

Syötteet:

```
3
```

Tuloste:

```
Tämä ohjelma laskee Neperin luvulle desimaaleja.
Anna kierrosten määrä: 3
Neperin luvun likiarvo annetulla kierrosten määrällä:
2.5
Kiitos ohjelman käytöstä.
```

Esimerkkiajo 2

Syötteet:

```
10
```

Tuloste:

```
Tämä ohjelma laskee Neperin luvulle desimaaleja.
Anna kierrosten määrä: 10
Neperin luvun likiarvo annetulla kierrosten määrällä:
2.7182815255731922477
Kiitos ohjelman käytöstä.
```

L11T2: Lukusarjat: Pell'n luvut (P)

Tehtävän taso: Perustaso

Pell'n luvut ovat matematiikassa päättymätön sarja kokonaislukuja. Kymmenen ensimmäistä Pell'n lukua ovat:

$$0, 1, 2, 5, 12, 29, 70, 169, 408, 985, \ldots$$

Pell'n luvut voidaan laskea alla olevalla kaavalla:

$$P_n = \begin{cases} 0 & \text{jos } n = 0 \\ 1 & \text{jos } n = 1 \\ 2P_{n-1} + P_{n-2} & \text{muutoin.} \end{cases}$$

Tee Python-ohjelma, joka laskee Pell'n lukuja yllä olevaa kaavaa käyttäen. Kysy käyttäjältä laskettavien lukujen määrä. Tulosta luvut allekkain alla olevan esimerkkiajon mukaisesti.

Jos käyttäjä antaa kierrosten määräksi alle 1, tulosta käyttäjälle alla oleva virheilmoitus:
`"Kierroksia on oltava vähintään 1 kpl."`

Esimerkkiajo 1

Syötteet:

```
5
```

Tuloste:

```
Tämä ohjelma laskee Pell'n lukusarjaa.
Anna laskettavien lukujen määrä: 5
5. ensimmäistä Pell'n lukua:
1. Pell'n luku: 0
2. Pell'n luku: 1
3. Pell'n luku: 2
4. Pell'n luku: 5
5. Pell'n luku: 12
Kiitos ohjelman käytöstä.
```

L11T3: Rekursiivinen algoritmi (P)

Tehtävän taso: Perustaso

Tee Python-ohjelma, joka laskee Fibonaccin lukujonon lukuja rekursiivisesti. Ohjelmaan kannattaa laittaa pääohjelma, joka kutsuu aliohjelmaa, joka hoitaa laskennan rekursion avulla. Tämä ohjelma on siis lyhyt ja yksinkertainen, mutta edellyttää rekursio-idean ymmärtämistä ja toteuttamista. Kysy käyttäjältä luku, jolle Fibonaccin luku lasketaan. Tätä lukua kannattaa käyttää rekursion ytimenä.

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

Esimerkkiajo 1

Syötteet:

```
9
```

Tuloste:

```
Tämä ohjelma laskee Fibonaccin luvun rekursiivisesti.
Anna luku, jolle Fibonaccin luku lasketaan: 9
Fibonaccin luku luvulle 9 on 55.
Kiitos ohjelman käytöstä.
```

L11T4: Nopea hakualgoritmi (T)

Tehtävän taso: Tavoitetaso

On olemassa vain kolme lukua, jotka voidaan kirjoittaa numeroidensa neljännen potenssin summana:

$$1634 = 1^4 + 6^4 + 3^4 + 4^4$$
$$8208 = 8^4 + 2^4 + 0^4 + 8^4$$

$$9474 = 9^4 + 4^4 + 7^4 + 4^4$$

(Koska $1 = 1^4$ *ei ole summa, sitä ei lasketa mukaan)*

Tehtävänäsi on tehdä annettuun ohjelmatiedostoon hakufunktio, joka testaa pitkästä lukujoukosta (lähes 200 000 lukua), onko siinä sellaisia lukuja, jotka ovat numeroidensa **viidennen** potenssin summa.

Tällä kertaa haemme ratkaisua, joka on mahdollisimman nopea. Siksi ratkaisulta vaaditaan, että se antaa tiedostoissa L11T4D1.txt, L11T4D2.txt, L11T4D3.txt sekä L11T4D4.txt olevista luvuista oikean vastauksen (löytyy ja numerot tai ei löydy) alle 2 sekunnissa. Numerot ovat datajoukossa satunnaisessa järjestyksessä, jolloin oikeat luvut voivat olla missä tahansa kohtaa tiedostoa. Kun hakualgoritmisi löytää vastauksen, tallenna luku annettuun listaan. Tehtävä liittyy ohjelmien suoritustehokkuuteen eikä sille ole olemassa yhtä oikeaa ratkaisua. Moodlessa L11 tehtävien tiedostoissa on Python ohjelman runko `L11T4Runko.py`, ota tämä tiedosto lähtökohdaksi ja lisää aliohjelmaan hakufunktio tarvittava koodi tiedoston lukemiseen ja oikeiden lukujen etsimiseen.

Palauta tekemäsi ratkaisu CodeGradeen normaalisti kokonaisena tiedostona. Muista lisätä otsikkotiedot (tekijä, op numero, ...) normaaliin tapaan.

Esimerkki syötetiedostosta 'L11T4D1.txt':

```
4148
4145
4155
4157
4156
4154
4159
4147
4152
4146
4151
4158
4153
4149
4150
```

Esimerkkiajo 1

Syötteet:

```
L11T4D4.txt
```

Tuloste:

```
Anna luettavan tiedoston nimi: L11T4D4.txt
Hakualgoritmi ei löytänyt sopivia lukuja.
Kiitos ohjelman käytöstä.
```

Esimerkkiajo 2

Syötteet:

```
L11T4D1.txt
```

Tuloste:

```
Anna luettavan tiedoston nimi: L11T4D1.txt
Hakualgoritmi oli riittävän nopea!
Se löysi 2 lukua:
4151, 4150,
Kiitos ohjelman käytöstä.
```