# 1. PostFix-laskin / symbolien tasapainotusohjelma

Tässä harjoitustyössä tehdään joko postfix-laskimen eri variaatioita tai symbolien tasapainotusohjelman eri variaatioita. Alla on kerrottu speksit variaatioille. Samoin alta löytyy lista palautukseen liittyvistä vaatimuksista

## 1.1. Postfix-laskin

Noudata työn tekemisessä järjestystä yhden pisteen työ -> kahden pisteen työ -> … eli esim. viiden pisteen arvoisen työn tulee sisältää tasojen 1 – 5 kaikki vaaditut toiminnot. Siten työtä kannattaa tehdä inkrementaalisesti aina vaihe kerrallaan. Siirtyessäsi tekemään seuraavan pisteen arvoista työtä älä ota mitään vanhoja ominaisuuksia pois työstäsi.

### 1.1.1. Yhden pisteen työn vaatimukset

Toteuta toimiva PostFix-laskin luennoilla läpikäytyjä ohjelmakoodeja soveltaen. Tähän versioon ei tarvitse tehdä mitään lisätoimintoja.

### 1.1.2. Kahden pisteen työn vaatimukset

Laskimen on toteutettava yhden pisteen vaatimukset ja lisäksi siihen tulee lisätä Vaihda-toiminto (esim. komento x (e**x**change)), joka vaihtaa ajon aikana laskimen käyttämässä pinossa kahden ylimmän alkion sisällöt keskenään. Esim. laskutoimitus voisi olla:

1 2 x –

, joka laskeekin 2 - 1:n.

### 1.1.3. Kolmen pisteen työn vaatimukset

Lisää laskimeen summa-toiminto (komento s (= **s**um)), joka laskee kaikkien pinossa sillä hetkellä olevien lukujen yhteissumman, poistaa tämän jälkeen pinosta summaamiseen käytetyt luvut ja lisää lopuksi yhteissumman pinon huipulle.

Esim.

1 2 3 s

:n jälkeen pinossa on vain yksi alkio, jonka arvo on 6.

### 1.1.4. Neljän pisteen työn vaatimukset

Toteuta laskimeesi keskiarvotoiminto, joka laskee kaikkien pinossa olevien lukujen keskiarvon. Keskiarvolaskentaan osallistuvat luvut poistetaan pinosta (kaikki) ja tulos (keskiarvo) painetaan pinon päälle. Nimeä tämä laskutoimitus kirjaimella a (= **a**verage).

### 1.1.5. Viiden pisteen työn vaatimukset

Tee laskimeesi Unix-/Linux-laitteilta löytyvän dc-ohjelman kaltainen käyttöliittymä, joka ei tulosta mitään kehoitteita vaan ohjelmalle annetaan suoraan haluttu laskulauseke postfix-muodossa ja ohjelma tulostaa sen arvon (ks. http://en.wikipedia.org/wiki/Dc_(computer_program)). Esimerkki:

1 3 + 7.5 –
-3.5

Viiden pisteen laskimessa tulee olla seuraavat laskutoimitukset: yhteenlasku (+), vähennyslasku (-), kertolasku (*), jakolasku (/), jakojäännösoperaatio (%), potenssiin korotus (^), neliöjuuri (v) sekä edellä vaaditut laskutoimitukset 1 – 4 pisteen arvoisissa töissä.

Koska mitään aiemmin tehtyä ei oteta pois laskinohjelmasta sisällytä kehoitteiden tulostaminen ohjelmaan  jos se käynnistetään tietyllä komentorivioptiolla; esim. laskin.exe –p
(= prompt).

## 1.2. Symbolien tasapainotusohjelma

Tämä on siis vaihtoehto postfix-laskimen tekemisellä. Tasapainotusohjelman osalta noudatetaan tekemisessä samoja periaatteita kuin laskimessa; esim. teet ensin yhden pisteen arvoisen sovelluksen ennen kuin lähdet tavoittelemaan kahden pisteen arvoista ratkaisua; jälkimmäisessä pitää olla mukana yhden pisteen työn ominaisuudet. Tämä pätee kauttaaltaan; esim. viiden pisteen arvoinen ohjelma sisältää kaikki alla vaaditut toiminnot.

### 1.1.6. Yhden pisteen työn vaatimukset

Luennoilla esitetty symbolien tasapainotusohjelma toteutetaan sillä lisäominaisuudella, että haluttaessa se lukee syötteen tiedostosta; esim. lähdekooditiedostosta Main.cpp.

### 1.1.7. Kahden pisteen työn vaatimukset

Jos ohjelmasi lukee syötteen tiedostosta se kaiutetaan päätteelle haluttaessa. Syötteessä olevasta virheestä annetaan virheilmoitus: sulkuja liikaa tai liian vähän, millaiselle alkusululle ei löytynyt vastinetta tai vastaavasti millainen loppusulku löytyi ilman vastaavaa alkusulkua.

### 1.1.8. Kolmen pisteen työn vaatimukset

Ohjelmasi tulee osata käsitellä merkkien, merkkijonojen ja kommenttien sisällä olevat sulkumerkit oikein: niitä ei oteta lainkaan lukuun tasapainotuksia tarkistettaessa. Esim. seuraavalla rivillä symbolit ovat tasapainossa:

```
( [       /* ( */                ”))))” ’[’                    ])
```

### 1.1.9. Neljän pisteen työn vaatimukset

Virheilmoitus tehdään informatiivisemmaksi siten, että virheen löytyessä siitä tulostetaan järkevä virheilmoitus sekä lisäksi virheellisen rivin järjestysnumero 1:sta alkaen ja koko virheellinen rivi (pyritään matkimaan kääntäjän toimintaa). Virheellisen rivin sisällön tulostuksen yhteydessä korostetaan virheen paikkaa rivillä alleviivauksella alla olevan esimerkin mukaisesti.

Syötetiedoston sisältö: if (( i == 0 )

Ohjelma tulostaa:

```
VIRHE:      ylimääräinen alkusulku
RIVINUMERO: 5
RIVI:       if (( i == 0 )
                -
```

### 1.1.10. Viiden pisteen työn vaatimukset

Lisää tasapainotusohjelmaasi vielä seuraavat ominaisuudet:

- myöskin pitkien kommenttien tasapaino tarkistetaan eli kutakin ”/*” –merkkiparia vasten pitää löytyä syötteestä vastaava kommentin lopettava ”*/” –merkkipari. Huomaa näiden kommenttimerkkien käsittely C++ -ohjelmointikielessä: ”/*”:in aloittava kommentti loppuu heti kun vastaava ”*/” löytyy; esim. sisäkkäisiä kommentteja ei voi tehdä, esim:

  ```
  /* lksjkdfsd fsdk /* ö öalkdjf  */ skljdfhsd */
  ```

  kommentin vaikutusalue näkyy harmaalla taustavärillä. Tasapainotusohjelman tulee osata kommenttien käsittely samalla tavoin kuin kääntäjä sen tekee; mekanismi on erilainen kuin sulkujen osalta jolloin sisäkkäisyys sallitaan.

- Myös merkkien ja merkkijonojen aloitus- ja lopetussymboleille tulee löytyä vastinparit:

  ```
  ”Hello, world!”                    ’c’
  ```

- template-mekanismiin liittyvien ’<’- ja ’>’ –merkkien tulee vastata toisiaan; ks. esim. STL-stack:in käsittelyyn tehdyn ohjelman ohjelmakoodia.

- huomioi edelleen se, että tasapainotettavat symbolit voivat olla merkkien, merkkijonojen, kommenttien sisällä jolloin ne jätetään pois tasapainotustarkastelusta

## 1.3. Vaatimuksia palautukselle

- palautetaan linkki viimeiseen committiin gitlabissa
- linkin takaa tulee löytyä myös README.DOCX –tiedosto, johon on dokumentoitu:
    - ohjelman käännös-/linkkaus-/asennus-/ajo-ohje niin, että testaaja pääsee testaamaan ohjelmaa seuraamalla näitä ohjeita
    - ohjelman jokaisen tason (1-5) yhteydessä ajetut testitapaukset, joilla ohjelman toiminnasta on varmistuttu. Käytännössä dokumentoi annettu syötte (cmd-ikkunasta / tiedostosta). Jos syötteenä on käytetty tiedostoa, jossa on isompi sisältö koko tiedosto pitää liittää projektiin mukaan ja se käyttö testauksessa pitää ohjeistaa (=> testaaja pystyy testaamaan helposti uudestaan testitapauksenne)
    - mitä pistemäärää tavoittelet palautetulla harjoitustyöllä
    - palauttajan yhteystiedot: sähköpostiosoite ja puhelinnumero
    - tuntikirjanpito työn tekemiseen kuluneesta ajasta tyyliin pvm, käytetty aika puolen tunnin tarkkuudella, mitä tuona aikana tehtiin. Tällaisia rivejä kannattaa kerätä esim. Excel-tiedostoon ( mukana projektissa) tai sitten rivit ovat README.DOCX –tiedostossa.
    - selitä lyhyesti sanallisesti kullakin tasolla käyttämäsi suunnitteluratkaisun. Eli mikä on kyseisellä tasolla tekemäsi ohjelmakoodin idea lyhyesti: miten ratkaisit ohjelmointiongelman sanoin selitettynä ? Tässä tietoa pitää tiivistää; jokaista ohjelmariviä ei pidä kertoa sanallisesti. Esim. voit kertoa jonkin algoritmin vaikka pseudokoodina tai ihan sanallisesti. Jos asian selittämistä auttaa kuva niin liitä sellainen mukaan palautukseen (jonkinlainen UML-kaavio tms.)

README.DOCX-tiedoston ulkoasun on oltava TAMKin kirjallisten töiden raportointiohjeen mukainen (löydät tämän intran hakutoiminnolla; tähän on tehty valmis Word-template).