Rekistereiden käyttö

Nykyaikaiset tietokoneen suorittimet sisältävät rajallisen määrän *rekistereitä* — yleiskäyttöisiä muistipaikkoja jotka ovat merkittävästi nopeampia kuin keskusmuisti. Laskutoimitukset (kuten yhteen- ja kertolasku ynnä muut) odottavat saavansa argumentit rekistereistä ja palauttavat vastauksen johonkin rekisteriin.

Tässä tehtävässä tarkastellaan *rekistereiden käytön* ongelmaa lauseketta laskettaessa. Kääntäjässä lauseke esitetään sisäisesti puuna. Puun lehtisolmut vastaavat arvoja jotka on noudettava keskusmuistista. Sisäsolmut (eli muut kuin lehdet) vastaavat laskutoimituksia ja jokaisella on yhtä monta lasta kuin laskutoimituksella on argumentteja. Kaikkien argumenttien arvot täytyy tietenkin olla laskettu ennen kuin itse laskutoimitus voidaan suorittaa.

Koska rekistereitä on vain rajallinen määrä, kääntäjän täytyy päättää mitkä välitulokset voidaan säilyttää rekistereissä (joissa ne ovat välittömästi valmiina kun niitä tarvitaan) ja mitkä pitää tallettaa välillä keskusmuistiin (josta ne pitää ensin ladata takaisin kun niitä tarvitaan). Voi olla myös hyödyllistä muuttaa laskutoimituksen argumenttien laskentajärjestystä (tämän vuoksi useimmat korkean tason ohjelmointikielet eivät takaa mitään tiettyä järjestystä).

Tehtävänäsi on kirjoittaa sellainen ohjelma, joka saa syötteekseen lausekkeen puumuodossa, ja etsii sille sellaisen rekistereiden käyttösuunnitelman ja laskentajärjestyksen jonka hinta on mahdollisimman pieni.

Syöttö. Syötetiedoston `REGS.IN` ensimmäinen rivi sisältää rekistereiden lukumäärän N ($1 \leq N \leq 100$). Toinen rivi sisältää kaksi kokonaislukua: hinnan sille kun arvo ladataan keskusmuistista rekisteriin, C_l ($1 \leq C_l \leq 100$), ja hinnan sille kun arvo talletetaan rekisteristä keskusmuistiin, C_s ($1 \leq C_s \leq 100$). Loput tiedostosta kuvaa annetun lausekkeen puumuodossa sen juurisolmusta alkaen:

- puun kuvauksen ensimmäinen rivi sisältää solmun lasten lukumäärän, K_x ($0 \leq K_x \leq 10$ ja $K_x \leq N$);
- jos $K_x = 0$, niin tämä on lehtisolmu ja tämän alipuun kuvaus loppuu;
- jos $K_x > 0$, niin tämä on sisäsolmu, ja:
  - seuraava rivi sisältää yhden kokonaisluvun: sen operaation hinnan jota tämä solmu esittää, C_x ($1 \leq C_x \leq 100$);
  - sen jälkeen annetaan tämän solmun alipuut K_x kuvauksena tällä samalla tavalla.

Puun solmut on numeroitu luvusta 1 lukuun M siinä järjestyksessä missä ne esiintyvät syötetiedostossa. Voidaan olettaa että $M \leq 10\,000$.

Tulostus. Tulostiedoston `REGS.OUT` ensimmäisellä rivillä pitää olla halvin hinta jolla syötteenä annetun lausekkeen arvo voidaan laskea. Sen rivin jälkeen pitää tiedostossa olla yksi rivi jokaiselle lausekepuun solmulle eikä muuta. Solmua vastaavalla rivillä pitää olla kaksi kokonaislukua: ensimmäinen rivin luvuista on kyseisen solmun järjestysnumero, jälkimmäinen rivin luvuista on joko 1 jos solmun tulos pitää säilyttää rekisterissä tai 0 jos se pitää tallettaa keskusmuistiin (jolloin vastaava kustannus C_s on lisätty myös kokonaishintaan).

Laskutoimitukset pitää luetella siinä järjestyksessä jossa ne on suoritettava jotta lausekkeen arvon laskennan kokonaishinta on pienin mahdollinen ja noudattaen seuraavia rajoituksia:

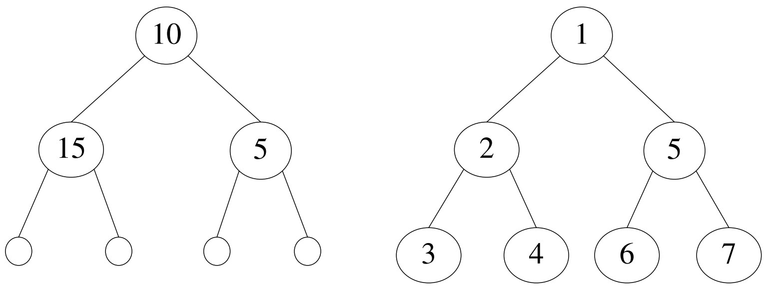
- solmu voidaan suorittaa vasta sen jälkeen kun kaikki sen lapset on suoritettu;
- kaikki ne suoritettavan laskutoimituksen argumentit, jotka eivät ole rekistereissä, täytyy ladata keskusmuistista (hintaan C_l arvoa kohden);
- juuri suoritetun laskutoimituksen argumenttien rekisterit voidaan käyttää heti uudelleen, erityisesti juuri tämän saman laskutoimituksen vastauksen tallentamiseen.

Jos on useita eri suunnitelmia joilla voi saavuttaa saman minimihinnan, voit tulostaa niistä minkä tahansa.

Esimerkki.

REGS.IN	REGS.OUT
2	47
3 2	2 0
2	5 1
10	1 1
2	
15	
0	
0	
2	
5	
0	
0	

Huomautus. Allaoleva kuva esittää ylläolevaa esimerkkisyötetiedostoa: molemmat puut vastaavat lausekkeen puumuotoa siten, että vasemmanpuoleinen puu näyttää sisäsolmujen kustannukset ja oikeanpuoleinen puu näyttää solmujen järjestysnumeroinnin.



Ylläolevan tulostiedoston mukaiselle arvonlaskentasuunnitelmalle saadaan kokonaishinnaksi

$$(C_l + C_l + 15 + C_s) + (C_l + C_l + 5) + (C_l + 10) = 47.$$

Huomautus. Sinulle annetaan ohjelma nimeltä `REGSCHECK` joka tarkistaa vastaako tulostiedosto `REGS.OUT` syötetiedostoa `REGS.IN` (mutta ei sitä onko sen ehdottama hinta halvin mahdollinen) ja antaa selventäviä virheilmoituksia jos tiedostot eivät vastaa toisiaan.