# Flash-muisti

**1 sec / 10 sec** **1 GB**

Mikrokontrollerin sirussa on $B$ bittiä sisäistä flash-muistia. Meidän täytyy tallentaa ja päivittää $M$-bitin muuttujaa muistissa. Flash-muistin rajoituksena on, että minkä tahansa yksittäisen 0-bitin voi vaihtaa 1-bitiksi, mutta vaihtaminen 1-bitistä 0-bitiksi on mahdollista vain tyhjentämällä koko muisti. Muisti voidaan tyhjentää vain rajoitetun määrän kertoja, ennen kuin siru on kulunut loppuun ja se täytyy vaihtaa. Siksi on toivottavaa pystyä kirjoittamaan mahdollisimman monta arvoa ennen tyhjentämistä.

Tehtäväsi on kehittää tehokas tapa tallentaa arvoja flash-muistiin niin, että nykyinen arvo voidaan aina hakea. Tarkemmin ohjelmasi tulee toteuttaa kaksi operaatiota:

- Kirjoita arvo: Syötteenä on muistin nykyinen tila sekä uusi kirjoitettava arvo, ja sinun tulee tulostaa muistin uusi tila.
- Lue arvo: Syötteenä on muistin tila joidenkin kirjoitusoperaatioiden jälkeen, ja sinun tulee tulostaa viimeksi kirjoitettu arvo.

Luku- ja kirjoitusoperaatiot eivät voi välittää keskenään tietoa mitenkään muuten kuin, että ne voivat lukea muistin nykyisen tilan syötteestä sekä kirjoitusoperaatio voi vaihtaa joitakin bittejä (mahdollisesti ei mitään) 0-bitistä 1-bitiksi ennen uuden tilan tulostamista.

**Keskustelu.** Tämä on interaktiivinen tehtävä. Kun ohjelmasi käynnistetään, ensimmäisellä rivillä on kokonaisluku $T$, missä $T = 0$ tarkoittaa, että ohjelma kirjoittaa arvoja muistiin, ja $T = 1$ tarkoittaa, että se lukee arvoja muistista. Seuraavalla rivillä on kokonaisluvut $B$ ja $M$. Seuraavat rivit kuvailevat operaatiot.

Sekä kirjoittamisessa että lukemisessa jokaisen operaation ensimmäisellä rivillä on kokonaisluku $C$, missä $C = 0$ tarkoittaa, että ei tule enempää pyyntöjä ja ohjelman tulee sulkeutua, mutta $C = 1$ tarkoittaa, että ohjelman tulee jatkaa toimintaa.

- Kun $T = 0$ ja $C = 1$, toisella rivillä on kaksi välilyönneillä erotettua merkkijonoa: muistin nykyistä tilaa kuvaava $B$ bitin jono sekä uutta kirjoitettavaa arvoa kuvaava $M$ bitin jono. Jos ohjelmasi pystyy kirjoittamaan uuden arvon muistiin vain muuttamalla joitain 0-bittejä 1-biteiksi, sen tulee tulostaa ensin kokonaisluku 1 ja seuraavalle riville $B$ bittiä, jotka kuvaavat muistin uuden tilan. Jos ohjelmasi ei pysty kirjoittamaan uutta arvoa muistiin, sen tulee tulostaa kokonaisluku 0.
- Kun $T = 1$ ja $C = 1$, toisella rivillä on yksi merkkijono, muistin tila $B$ bitin jonona, ja ohjelmasi tulee tulostaa yksi merkkijono: $M$ bitin jono, joka on viimeksi muistiin kirjoitettu arvo.

**Esimerkki.**

| Syöte | Tuloste |
|---|---|
| `0` | |
| `6 2` | |
| `1` | |
| `111111 00` | |
| | `0` |
| `1` | |
| `000000 11` | |
| | `1` |
| | `110000` |
| `0` | |

Tässä esimerkissä ohjelmasi tulee kirjoittaa 2-bittisiä arvoja 6-bitin muistiin. Ensimmäinen pyyntö on kirjoittaa arvo 00, mitä ohjelmasi ei pysty tekemään. Seuraava pyyntö on kirjoit-

taa arvo 11, minkä ohjelmasi pystyy tekemään. Huomaa, että muistin nykyinen tila toisessa pyynnössä ei ole sama kuin ohjelmasi tuloste ensimmäisen pyynnön jälkeen.

**Esimerkki.**

| Syöte | Tuloste |
|---|---|
| 1<br>6 2<br>1<br>110000 | |
| | 11 |
| 1<br>110100 | |
| | 01 |
| 0 | |

Tässä esimerkissä ohjelmasi tulee lukea 2-bittisiä arvoja 6-bitin muistista. Ensimmäinen pyyntö on lukea muistin tilasta 110000, josta ohjelmasi lukee arvon 11. Toinen pyyntö on lukea muistin tilasta 110100, josta ohjelmasi lukee arvon 01.

**Huomautus.** Jotta ohjelmasi arvostellaan varmasti oikein, sinun tulee käyttää flush-operaatiota jokaisen pyynnön jälkeen (huomaa myös, että tuloste päättyy aina rivinvaihtoon):

| Kieli | Komento |
|---|---|
| C | `fprintf(stdout, "1\n%s\n", s);`<br>`fflush(stdout);` |
| C++ | `cout << 1 << "\n" << s << endl;` |
| Java | `System.out.println("1");`<br>`System.out.println(s);`<br>`System.out.flush();` |
| Python | `sys.stdout.write("1\n{0}\n".format(s))`<br>`sys.stdout.flush()` |

**Testaus.** Jokaisessa tapauksessa ohjelmastasi käynnistetään 4 instanssia, 2 lukemista varten ja 2 tulostamista varten. Suoritusaika ja muistiraja ovat yhteiset näille instansseille. **Jos koetat välittää tietoa näiden instanssien välillä, tämä katsotaan huijaamiseksi ja suorituksesi hylätään.**

Muistista alustetaan nollaksi useita osuuksia, joista jokainen on $B$ bitin pituinen. Kirjoitus- ja tulostusoperaatiot suoritetaan jossain kelvollisessa järjestyksessä.

Kirjoitusoperaatiossa instanssille annetaan jonkin muistilohkon nykyinen tila ja siihen kirjoitettava arvo. Voit olettaa, että kirjoitettavat arvot valitaan tasaisesta satunnaisjakaumasta väliltä $0 \ldots 2^M - 1$ riippumattomasti kaikesta muusta. Jos ohjelmasi pystyy kirjoittamaan arvon, muistilohkon tilaksi asetetaan ohjelmasi antama tila. Jos ohjelmasi ei pysty kirjoittamaan arvoa, tätä muistilohkoa ei käytetä missään tulevassa kirjoitusoperaatiossa.

Lukuoperaatiossa instanssille annetaan muistilohkon tila onnistuneen kirjoitusoperaation jälkeen. Tässä tarkastetaan, että ohjelmasi antama arvo on sama, kuin mitä kirjoitusoperaation aikana tuli kirjoittaa. Lukuoperaatio suoritetaan tarkalleen kerran jokaisen onnistuneen kirjoitusoperaation tulosteelle.

**Arvostelu.** Jokaisessa testiryhmässä ohjelmasi saa pisteitä suhteessa keskimääräiseen kirjoitettujen arvojen määrään ryhmän testeissä. Tarkemmin sanoen jos ohjelmasi pystyy kirjoittamaan keskimäärin $V$ arvoa muistilohkoa kohden, sen pistemäärä on $100 \cdot V/P$ % testiryhmän arvosta,

missä $P$ on annettu alhaalla. Jos ohjelmasi antaa väärän arvon missä tahansa lukuoperaatiossa, koko ryhmän pisteet jäävät nollaan. Jos tulee jokin muu virhe, kyseisessä testissä luettujen arvojen määrä lasketaan nollana.

Testiryhmät täyttävät seuraavat ehdot:

1. (5 pistettä) $B = 16$, $M = 8$, $P = 4.062445024495069624056$.
2. (5 pistettä) $B = 32$, $M = 8$, $P = 12.2649048413009648341 77$.
3. (5 pistettä) $B = 32$, $M = 16$, $P = 4.129591513707784802006$.
4. (5 pistettä) $B = 64$, $M = 8$, $P = 30.039277894268828900030$.
5. (5 pistettä) $B = 64$, $M = 16$, $P = 12.953148094217360432715$.
6. (5 pistettä) $B = 64$, $M = 32$, $P = 4.073559788233661501537$.
7. (5 pistettä) $B = 128$, $M = 8$, $P = 69.7778922289287475487 75$.
8. (5 pistettä) $B = 128$, $M = 16$, $P = 34.7317912751436352409 76$.
9. (5 pistettä) $B = 128$, $M = 32$, $P = 13.9507889877056389086 63$.
10. (5 pistettä) $B = 128$, $M = 64$, $P = 4.039918210604800133907$.
11. (5 pistettä) $B = 256$, $M = 8$, $P = 174.4680470860710385114 53$.
12. (5 pistettä) $B = 256$, $M = 16$, $P = 82.2226141514041773345 54$.
13. (5 pistettä) $B = 256$, $M = 32$, $P = 37.6293822697692064889 16$.
14. (5 pistettä) $B = 256$, $M = 64$, $P = 14.263462282054140577686$.
15. (5 pistettä) $B = 256$, $M = 128$, $P = 4.015569093893943430859$.
16. (5 pistettä) $B = 512$, $M = 16$, $P = 204.7462421274103461702 21$.
17. (5 pistettä) $B = 512$, $M = 32$, $P = 91.7785951480731115398 47$.
18. (5 pistettä) $B = 512$, $M = 64$, $P = 39.230279242145938712621$.
19. (5 pistettä) $B = 512$, $M = 128$, $P = 15.000000002167672268601$.
20. (5 pistettä) $B = 512$, $M = 256$, $P = 4.005423277111055468876$.

Lisäksi kaikissa testitapauksissa $N \cdot M \leq 120000$, jossa $N$ on suurin sallittu kirjoitusoperaatioiden määrä, jotka ohjelmasi täytyy pystyä tekemään.

Kilpailun aikana ratkaisusi arvostellaan pienellä joukolla testejä joka ryhmästä. Kilpailun jälkeen viimeinen lähetyksesi ja parhaan pistemäärän pienestä testiaineistosta saanut lähetys arvostellaan laajalla testiaineistolla, ja näistä parempi tulee olemaan lopullinen pistemääräsi tehtävästä. **Kilpailujärjestämässä näkemäsi pistemäärä ei ole lopullinen pistemääräsi.** Tämän tavoitteena on kasvattaa pisteytyksen tarkkuutta. Arvostelu laajalla testiaineistolla kilpailun aikana olisi liian hidasta.