Zibatmiņa

1 s / 10 s **1 GB**

Kādam mikrokontroliera čipam ir B iebūvētās zibatmiņas biti. Mums šajā atmiņā ir jāsaglabā un jāatjauno M bitu mainīgā vērtība. Zibatmiņai ir sevišķa īpašība: atsevišķus bitus var mainīt no `0` uz `1`, bet bita nomaiņa no `1` uz `0` ir iespējama, tikai izdzēšot visu atmiņu. Atmiņu var dzēst ierobežotu skaitu reižu, pirms čipa resurss tiek izlietots un čips ir jāmaina. Secīgi, ir vēlams varēt ierakstīt pēc iespējas vairāk vērtību pirms kārtējās atmiņas dzēšanas.

Jūsu uzdevums ir izdomāt efektīvu veidu, kā saglabāt mainīgā vērtības zibatmiņā tā, ka tagadējo vērtību vienmēr var nolasīt. Precīzāk, jūsu programmai ir jārealizē divas operācijas:

- Vērtības rakstīšana: šī operācija ievadā saņem tagadējo atmiņas stāvokli un jaunu mainīgā vērtību, kas ir jāieraksta, un jums ir jāizvada jauns atmiņas stāvoklis.
- Vērtības lasīšana: šī operācija ievadā saņem atmiņas stāvokli pēc kāda rakstīšanas operāciju skaita, un jums ir jāizvada pēdējās rakstīšanas operācijas laikā ierakstītā vērtība.

Jūsu lasīšanas un rakstīšanas operācijas nevar savā starpā apmainīties ar informāciju jebkurā veidā, izņemot nolasot atmiņas tagadējo stāvokli no ievada un rakstīšanas operācijai nomainot dažus bitus (iespējams, nevienu) no `0` uz `1` pirms jaunā stāvokļa izvadīšanas.

Komunikācija. Šis ir interaktīvs uzdevums. Jūsu programmas darbības sākumā pirmā ievada rinda satur veselu skaitli T, kur $T = 0$ nozīmē, ka jūs programma rakstīs vērtības atmiņā, un $T = 1$ nozīmē, ka tai būs jālasa vērtības no atmiņas. Otrā rinda satur veselus skaitļus B un M. Sekojošās rindas apraksta operācijas.

Gan rakstīšanai, gan lasīšanai katras operācijas pirmā rinda satur veselu skaitli C, kur $C = 0$ nozīmē, ka pieprasījumu vairs nebūs un jūsu programmai ir jābeidz darbība, bet $C = 1$ nozīmē, ka jūsu programmai ir jāturpina darbība.

- Kad $T = 0$ un $C = 1$, otrā rinda satur divas ar atstarpi atdalītas virknes: tagadējo atmiņas stāvokli kā virkni no B bitiem, kā arī jaunu vērtību, kas ir jāieraksta, kā virkni no M bitiem. Ja jūsu programma spēj ierakstīt jaunu vērtību atmiņā, tikai nomainot dažus bitus no `0` uz `1`, tai sākumā ir jāizvada vesels skaitlis 1 un nākamajā rindā jāizvada jauns atmiņas stāvoklis kā virkni no B bitiem. Ja jūsu programma nespēj ierakstīt jaunu vērtību atmiņā, tai ir jāizvada vesels skaitlis 0.
- Kad $T = 1$ un $C = 1$, otrā rinda satur atmiņas stāvokli kā virkni no B bitiem, un jūsu programmai ir jāizvada virkne no M bitiem — pēdējā atmiņā ierakstītā vērtība.

Piemērs.

Ievaddati	Izvaddati
`0`	
`6 2`	
`1`	
`111111 00`	
	`0`
`1`	
`000000 11`	
	`1`
	`110000`
`0`	

Šajā piemērā jūsu programma tiek palaista, lai rakstītu 2 bitu vērtības 6 bitu atmiņā. Pirmais pieprasījums ir ierakstīt vērtību `00`, ko jūsu programma nespēj izdarīt. Otrais pieprasījums ir

ierakstīt vērtību `11`, ko jūsu programma spēj izdarīt. Ievērojiet, ka atmiņas tagadējais stāvoklis otrajam pieprasījumam nesakrīt ar jūsu programmas izvadu pēc pirmā pieprasījuma.

Piemērs.

Ievaddati	Izvaddati

`1`
`6 2`
`1`
`110000`	

	`11`
`1` `110100`	
	`01`
`0`	

Šajā piemērā jūsu programma tiek palaista, lai lasītu 2 bitu vērtības no 6 bitu atmiņas. Pirmais pieprasījums ir nolasīt datus no atmiņas stāvokļa `110000`, no kā jūsu programma nolasa vērtību `11`. Otrais pieprasījums ir nolasīt datus no atmiņas stāvokļa `110100`, no kā jūsu programma nolasa vērtību `01`.

Piezīme. Lai nodrošinātu saziņu ar testēšanas vidi, jums ir jāsinhronizē (*flush*) izvada plūsma pēc katras atbildes (ievērojiet arī, ka izvads vienmēr beidzas ar jaunās rindas simbolu):

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

Testēšana. Katram testam vienlaicīgi tiks palaistas 4 jūsu programmas instances, 2 rakstīšanai un 2 lasīšanai. Izpildes laika un atmiņas ierobežojumi tiek piemēroti visām šīm instancēm kopā. **Jebkurš apzināts mēģinājums apmainīties ar datiem starp šīm instancēm ārpus protokola tiek uzskatīts par pārkāpumu un būs pamats diskvalifikācijai.**

Vairāki atmiņas apgabali, katrs B bitus garš, būs inicializēti ar nullēm. Pēc tam rakstīšanas un lasīšanas operācijas tiks izpildītas kādā iespējamā secībā.

Rakstīšanas operācijas laikā, rakstīšanas instance saņem tagadējo stāvokli vienam no apgabaliem un vērtību, kas tajā ir jāieraksta. Jūs varat pieņemt, ka ierakstāmās vērtības tika nejauši un vienmērīgi izvelētas no intervāla $0 \dots 2^M - 1$, neatkarīgi no visiem citiem faktoriem. Ja jūsu programma spēj ierakstīt mainīgā vērtību, apgabala stāvoklis tiks uzstādīts uz tādu, ko atgriež jūsu programma. Ja jūsu programma nespēj ierakstīt mainīgā vērtību, šis apgabals vairs netiks lietots turpmākajās rakstīšanas operācijās.

Lasīšanas operācijas laikā, lasīšanas instance saņem apgabala stāvokli pēc sekmīgas rakstīšanas operācijas. Pēc tam tiek pārbaudīts, ka vērtība, ko atgriež jūsu programma, sakrīt ar to vērtību, kurai bija jābūt ierakstītai rakstīšanas operācijas laikā. Lasīšanas operācija tiks izpildīta precīzi vienu reizi katram veiksmīgas rakstīšanas operācijas izvadam.

Vērtēšana. Katrā testu grupā jūsu programmas vērtējums ir proporcionāls vidējam vērtību skaitam, kas ticis ierakstīts šīs grupas testu izpildes laikā. Precīzāk, ja jūsu programma spēj vidēji uz vienu apgabalu ierakstīt V vērtības, tad tā iegūs $100 \cdot V/P$ procentus no testu grupas pilnajiem punktiem, kur P vērtība ir dota zemāk. Kad jūsu programma atgriež nepareizu vērtību jebkurā lasīšanas operācijā, vērtējums ir nulle visai grupai kopā. Jebkuram citam nesekmīgam rezultātam, attiecīgajā testā nolasīto vērtību skaits tiks skaitīts kā nulle.

Testu grupas atbilst sekojošiem nosacījumiem:

1. (5 punkti) $B = 16$, $M = 8$, $P = 4{,}062445024495069624056$.
2. (5 punkti) $B = 32$, $M = 8$, $P = 12{,}264904841300964834177$.
3. (5 punkti) $B = 32$, $M = 16$, $P = 4{,}129591513707784802006$.
4. (5 punkti) $B = 64$, $M = 8$, $P = 30{,}039277894268828900030$.
5. (5 punkti) $B = 64$, $M = 16$, $P = 12{,}953148094217360432715$.
6. (5 punkti) $B = 64$, $M = 32$, $P = 4{,}073559788233661501537$.
7. (5 punkti) $B = 128$, $M = 8$, $P = 69{,}777892228928747548775$.
8. (5 punkti) $B = 128$, $M = 16$, $P = 34{,}731791275143635240976$.
9. (5 punkti) $B = 128$, $M = 32$, $P = 13{,}950788987705638908663$.
10. (5 punkti) $B = 128$, $M = 64$, $P = 4{,}039918210604800133907$.
11. (5 punkti) $B = 256$, $M = 8$, $P = 174{,}468047086071038511453$.
12. (5 punkti) $B = 256$, $M = 16$, $P = 82{,}222614151404177334554$.
13. (5 punkti) $B = 256$, $M = 32$, $P = 37{,}629382269769206488916$.
14. (5 punkti) $B = 256$, $M = 64$, $P = 14{,}263462282054140577686$.
15. (5 punkti) $B = 256$, $M = 128$, $P = 4{,}015569093893943430859$.
16. (5 punkti) $B = 512$, $M = 16$, $P = 204{,}746242127410346170221$.
17. (5 punkti) $B = 512$, $M = 32$, $P = 91{,}778595148073111539847$.
18. (5 punkti) $B = 512$, $M = 64$, $P = 39{,}230279242145938712621$.
19. (5 punkti) $B = 512$, $M = 128$, $P = 15{,}000000002167672268601$.
20. (5 punkti) $B = 512$, $M = 256$, $P = 4{,}005423277111055468876$.

Visiem testiem izpildās arī $N \cdot M \leq 120\,000$, kur N ir lielākais rakstīšanas operāciju skaits, ko jūsu programmai varētu prasīt izpildīt.

Sacensību laikā jūsu risinājumu vērtēs uz neliela testu skaita no katras testu grupas. Pēc sacensībām, jūsu pēdējais iesūtījums un iesūtījums, kas ieguva vislielāko punktu skaitu uz šī mazā testu komplekta, tiks testēti uz lielāka testu komplekta, un labākais rezultāts starp šiem diviem iesūtījumiem būs jūsu gala vērtējums šajā uzdevumā. **Vērtējums, ko jūs redzēsiet testēšanas sistēmā, nav jūsu gala vērtējums.** Šīs metodes mērķis ir palielināt jūsu vērtējuma precizitāti. Testēšana uz pilna testu komplekta sacensību laikā būtu pārāk lēna.