Flashminne

1 sec / 10 sec **1 GB**

En mikrokontroller har B bitar inbyggt flashminne. Vi behöver spara en M-bits variabel i detta minne. Flashminne har den begränsningen att individuella bitar kan ändras från `0` till `1`, men för att gå från `1` till `0` behöver hela minnet nollställas. Minnet kan bara nollställas ett begränsat antal gånger innan chipet slits ut och behöver bytas. På grund av detta är det önskvärt att kunna uppdatera variabeln så många gånger som möjligt utan att nollställa.

Ditt uppdrag är att hitta på ett effektivt sätt för att lagra data i flashminnet så att det senaste skrivna värdet alltid kan återhämtas. Detta innebär att du behöver implementera följande två operationer:

- Skrivning: Indata för denna operation är minnets nuvärande tillstånd och det nya värdet som ska sparas. Utdata är det nya tillståndet för minnet.
- Läsning: Indata för denna operation är minnets tillstånd efter ett visst antal skrivningar. Utdata är det senaste värdet som sparades genom en skrivoperation.

Dina skriv och läsrutiner får inte utbyta information på något annat sätt än genom minnestillståndet.

Interaktion. Detta är en interaktiv uppgift. När ditt program startar kommer den första raden av indatan att innehålla ett heltal T, där $T = 0$ betyder att ditt program kommer att skriva värden till minnet och $T = 1$ betyder att det kommer läsa värden från minnet. Den andra raden innehåller hetalen B och M. Följande rader kommer beskriva operationerna.

Oavsett operation, kommer den första raden inehålla ett heltal C, där $C = 0$ betyder att det inte kommer några fler operationer och att ditt program bör avlutas, men $C = 1$ betyder att ditt program ska fortsätta.

- För $T = 0$ och $C = 1$ kommer den andra raden innehålla två strängar separerade med mellanslag. Dessa innhåller, respektive, det nuvärande tillståndet för minnet som en sekvens av B bitar och det nya värdet som ska skrivas som en sekvens av M bitar. Om ditt program kan skriva det nya värdet till minnet genom att bara ändra några bitar från `0` till `1` ska det först skriva ut talet 1 följt av det nya minnestillståndet som en sekvens av B biter. Annars, skriv 0.
- För $T = 1$ and $C = 1$ kommer den andra raden innehålla en sträng: det nuvarande tillståndet på minnet som en sekvens av B bitar och ditt program ska skriva ut det senaste skriva värdet som en sekvens av M bitar.

Exempel.

Indata	Utdata

`0`
`6 2`
`1`
`111111 00`	

	`0`
`1` `000000 11`	
	`1` `110000`
`0`	

I detta exepel startades ditt program för att skriva 2-bit värden till ett 6-bitars minne. Den första operationen var att skriva värdet `00`, vilket ditt program inte lyckades göra. Den andra operationen var att skriva värdet `11`, vilket ditt program lyckades med. Notera att minnestillståndet till den andra operationen inte stämmer överens med tillståndet efter den första operationen.

Exempel.

Indata	Utdata
1 6 2 1 110000	
	11
1 110100	
	01
0	

I detta exepel startades ditt program för att läsa 2-bit värden till ett 6-bitars minne. Den första operationen var att läsa från minnestillståndet `110000`, varifrån ditt program extraherade värdet `11`. Den andra operationen var att läsa från minnestillståndet `110100`, varifrån ditt program extraherade värdet `01`.

Anmärkning. Kom ihåg att spola efter dig!

Språk	Kommando
C	`fprintf(stdout, "1\n%s\n", s);` `fflush(stdout);`
C++	`cout << 1 << "\n" << s << endl;`

Java	`System.out.println("1");`
	`System.out.println(s);`
	`System.out.flush();`

Python	`sys.stdout.write("1\n{0}\n".format(s))` `sys.stdout.flush()`

Testning. I varje testfall kommer vi starta 4 instanser av ditt program samtidigt varav hälften läser och hälften skriver. Resursanvändningen för dessa instanser kommer att adderas **Det är inte tillåtet att försöka kommunicera utan att använda minnestillståndet för flashminnet!**

Flera minnesblock av längd B bitar kommer initialiseras med nollor. Sedan kommer skriv och läsoperationer att genomföras i någon godtycklig ordning.

Under en skrivoperation kommer en skrivare ges det nuvarande minnestillståndet för ett av minnesblocken och ett värde att skriva till det. Du kan anta att värderna som ska skrivas har blivit uniformt, slumpmässigt valda från intervallet $0 \dots 2^M - 1$ och är oberoende från varanda. Om ditt program kunde skriva värdet kommer blocket att uppdateras till det som returnerades av programmet. Om ditt program inte kunda skriva värdet kommer detta block inte användas mer.

Under en läsoperation kommer en läsare ges det nuvarande minnestillståndet för ett block efter en lyckad skrivoperation. Domaren ser till att värdet som returneras stämmer överens med det som skrevs senast. Varje värde som lyckas skrivas kommer läsas exakt en gång, och om du berättar för Jonas att du läst den här meningen får du en glass.

Bedömning. I varje testgrupp kommer ditt programs poäng vara proportionell mot det genomsnittliga antalet värden som skrevs av test i gruppen. Mer specifikt, om ditt program kan skriva V värden per block i genomsnitt kommer din poäng att vara $100 \cdot V/P\%$ av poängvärdet för testgruppen, där P ges nedanför. Om ditt program returnerar ett felaktigt värde i en läsoper-

ation kommer poängen för hela gruppen vara noll. För något annat fel kommer antalet värden som lästes i det testet räknas som noll.

Testgrupperna uppfyller följande begränsningar:

1. (5 poäng) $B = 16$, $M = 8$, $P = 4.062445024495069624056$.
2. (5 poäng) $B = 32$, $M = 8$, $P = 12.264904841300964834177$.
3. (5 poäng) $B = 32$, $M = 16$, $P = 4.129591513707784802006$.
4. (5 poäng) $B = 64$, $M = 8$, $P = 30.039277894268828900030$.
5. (5 poäng) $B = 64$, $M = 16$, $P = 12.953148094217360432715$.
6. (5 poäng) $B = 64$, $M = 32$, $P = 4.073559788233661501537$.
7. (5 poäng) $B = 128$, $M = 8$, $P = 69.777892228928747548775$.
8. (5 poäng) $B = 128$, $M = 16$, $P = 34.731791275143635240976$.
9. (5 poäng) $B = 128$, $M = 32$, $P = 13.950788987705638908663$.
10. (5 poäng) $B = 128$, $M = 64$, $P = 4.039918210604800133907$.
11. (5 poäng) $B = 256$, $M = 8$, $P = 174.468047086071038511453$.
12. (5 poäng) $B = 256$, $M = 16$, $P = 82.222614151404177334554$.
13. (5 poäng) $B = 256$, $M = 32$, $P = 37.629382269769206488916$.
14. (5 poäng) $B = 256$, $M = 64$, $P = 14.263462282054140577686$.
15. (5 poäng) $B = 256$, $M = 128$, $P = 4.015569093893943430859$.
16. (5 poäng) $B = 512$, $M = 16$, $P = 204.746242127410346170221$.
17. (5 poäng) $B = 512$, $M = 32$, $P = 91.778595148073111539847$.
18. (5 poäng) $B = 512$, $M = 64$, $P = 39.2302792421459387126 21$.
19. (5 poäng) $B = 512$, $M = 128$, $P = 15.000000002167672268601$.
20. (5 poäng) $B = 512$, $M = 256$, $P = 4.005423277111055468876$.

Utöver detta kommer alla testfall uppfylla $N \cdot M \leq 10^5$, där N är det maximala antalet skriv-operationer som ditt program kommer förväntas skriva.

Under tävlingen kommer din lösning att bedömas utifrån ett litet antal test i varje grupp. Efter tävlingen kommer din senaste submission och den submission som gav bäst poäng under tävlingen att testas på en större mängd testfall och de bästa av dessa två kommer vara din slutliga poäng på denna uppgift. Anledningen för detta är att förbättra noggrannheten.