Flashspeicher

1 sec / 10 sec **1 GB**

Ein Mikrocontroller hat B Bit an eingebautem Flashspeicher. Dort müssen wir eine M-Bit Variable speichern und updaten. Der Flashspeicher hat die Einschränkung, dass Bits von `0` zu `1` geändert werden können, jedoch ist das Ändern eines Wertes von `1` zu `0` nur möglich, indem man den gesamten Speicher auf `0` setzt. Der Speicher kann nur begrenzt oft komplett gelöscht werden, bevor der Chip ersetzt werden muss. Deshalb will man nur mit dem Überschreiben von `0`en mit `1`en auskommen.

Deine Aufgabe ist es, dir ein effizientes Verfahren auszudenken, sodass der aktuelle Wert der Variable immer abgefragt werden kann. Genauer muss dein Programm die folgenden Opreationen implementieren:

- Einen Wert schreiben: die Eingabe für diese Operation ist der aktuelle Zustand des Speichers und der Wert, der geschrieben werden soll. Die Ausgabe soll der neue Zustand des Speichers sein.
- Einen Wert lesen: die Eingabe für diese Operation ist der Zustand des Speichers nach einer oder mehr Schreiboperationen. Die Ausgabe soll der zuletzt geschriebene Wert sein.

Die Lese- und Schreiboperationen können keine Information untereinander austauschen, bis darauf, dass die Schreiboperation den aktuellen Zustand aus der Eingabe lesen und daraufhin Bits von `0` zu `1` ändern darf, bevor der neue Zustand in die Ausgabe geschrieben wird.

Interaktion. Dies ist eine interaktive Aufgabe. Die erste Zeile der Eingabe enthält einen Integer T, wobei $T = 0$ bedeutet, dass dein Programm Werte schreiben wird, und $T = 1$ bedeutet, dass dein Programm Werte lesen wird. Die zweite Zeile enthält die Integer B und M. Die folgenden Zeilen beschreiben die Operationen.

Für sowohl das Lesen wie auch das Schreiben enthält die erste Zeile einen Integer C. $C = 0$ bedeutet, dass keine weiteren Operationen kommen und das Programm sich beenden soll, $C = 1$ bedeutet, dass noch weitere Operationen kommen.

- Wenn $T = 0$ und $C = 1$, enthält die zweite Zeile zwei durch ein Leerzeichen getrennte Zeichenketten: den akuellen Zustand des Speichers als eine Folge von B Bits und den neuen zu schreibenden Wert als eine Folge von M Bits. Wenn dein Programm den neuen Wert in den Speicher schreiben kann, in dem es ausschließlich Werte von `0` auf `1` setzt, dann soll es zunächst den Integer 1 und in der nächsten Zeile den neuen Zustand des Speichers als Folge von B Bits ausgeben. Wenn dein Programm den neuen Wert nicht mehr in den Speicher schreiben kann, so soll 0 ausgegeben werden.
- Wenn $T = 1$ und $C = 1$, enthält die zweite Zeile eine einzelne Zeichenkette: den Zustand des Speichers als Folge von B Bits, und dein Programm soll eine einzelne Zeichenkette ausgeben: den zuletzt in den Speicher geschriebenen Wert als Folge von M Bits.

Beispiel.

Eingabe	Ausgabe
`0`	
`6 2`	
`1`	
`111111 00`	
	`0`
`1`	
`000000 11`	
	`1`
	`110000`
`0`	

In diesem Beispiel wird das Programm gestartet, um 2-bit Werte in einem 6-bit Speicher zu schreiben. In der ersten Operation soll der Wert `00` geschrieben werden, und das Programm ist nicht in der Lage dies zu tun. In der zweiten Operation soll der Wert `11` geschrieben werden, was das Programm schafft. Bemerke, dass der akuelle Zustand des Speichers für die zweite Operation nicht der Ausgabe der ersten Operation entspricht.

Beispiel.

Eingabe	Ausgabe

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

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

In diesem Beispiel wird das Programm gestartet, um 2-bit Werte aus einem 6-bit Speicher zu lesen. Die erste Operation bekommt den Zustand `110000`, aus welchem das Programm den Wert `11` liest. Die zweite Operation bekommt den Zustand `110100`, aus welchem das Programm den Wert `01` liest.

Anmerkung. Um sicher zu gehen, dass die Antworten an die Bewertungsumgebung geschickt werden, muss nach jeder Antwort der Ausgabebuffer geflushed werden (und die Ausgabe muss immer mit einer Newline enden):

Sprache	Befehl
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()`

Testvorgang. Für jeden Testfall werden 4 Instanzen deines Programms gleichzeitig gestartet, 2 die schreiben und 2 die lesen werden. Die CPU Zeit und das Arbeitsspeicherlimit ist für alle diese Instanzen zusammen. **Jeder Versuch, Daten über einen Nebenkanal auszutauschen wird als Betrug gewertet und führt zur Disqualifikation.**

Mehrere Blöcke des Speichers, alle B Bits lang, werden mit Nullen initialisiert. Daraufhin werden Schreib- und Leseoperationen in einer plausiblen Reihenfolge ausgeführt.

Während einer Schreiboperation wird einer Schreibinstanz der Zustand eines Blocks übergeben und ein Wert, der geschrieben werden soll. Du darfst annehmen, dass die Werte gleichverteilt zufällig zwischen 0 und $2^M - 1$ ausgewählt wurden und unabhängig von allem anderen sind. Wenn dein Programm in der Lage ist den Wert zu schreiben, dann wird der Zustand des Blocks auf die Ausgabe deines Programms gesetzt. Wenn dein Programm den Wert nicht schreiben kann, so wird nicht mehr auf den Block geschrieben.

Während einer Leseoperation wird einer Leseinstanz der Zustand nach einer erfolgreichen Schrei-

boperation übergeben. Es wird dann überprüft, ob der Wert, den das Programm zurückgibt, auch der Wert ist, der zuletzt in den Block hätte geschrieben werden sollen. Die Leseoperation wird genau einmal auf den Output von jeder erfolgreichen Schreiboperation ausgeführt.

Bewertung. In jeder Testgruppe ist die Punktezahl deines Programms proportional zur durchschnittlichen Anzahl an geschriebenen Werten in Tests dieser Gruppe. Genauer gesagt, wenn dein Programm die Variable im Schnitt V-mal updaten kann, ist der Score $100 \cdot V/P$ vom Wert der Testgruppe, wobei P unten gegeben ist. Wenn dein Programm einen falschen Wert in einer Leseoperation zurückgibt, ist die Punktezahl für die gesamte Gruppe Null. Für alle anderen Fehler wird die Zahl der im Test gelesenen Werte als 0 gezählt.

Die Testgruppen erfüllen die folgenden Bedingungen:

1. (5 points) $B = 16$, $M = 8$, $P \approx 4,06$.
2. (5 points) $B = 32$, $M = 8$, $P \approx 12,26$.
3. (5 points) $B = 32$, $M = 16$, $P \approx 4,12$.
4. (5 points) $B = 64$, $M = 8$, $P \approx 30,03$.
5. (5 points) $B = 64$, $M = 16$, $P \approx 12,95$.
6. (5 points) $B = 64$, $M = 32$, $P \approx 4,07$.
7. (5 points) $B = 128$, $M = 8$, $P \approx 69,77$.
8. (5 points) $B = 128$, $M = 16$, $P \approx 34,73$.
9. (5 points) $B = 128$, $M = 32$, $P \approx 13,95$.
10. (5 points) $B = 128$, $M = 64$, $P \approx 4,03$.
11. (5 points) $B = 256$, $M = 8$, $P \approx 174,46$.
12. (5 points) $B = 256$, $M = 16$, $P \approx 82,22$.
13. (5 points) $B = 256$, $M = 32$, $P \approx 37,62$.
14. (5 points) $B = 256$, $M = 64$, $P \approx 14,26$.
15. (5 points) $B = 256$, $M = 128$, $P \approx 4,01$.
16. (5 points) $B = 512$, $M = 16$, $P \approx 204,74$.
17. (5 points) $B = 512$, $M = 32$, $P \approx 91,77$.
18. (5 points) $B = 512$, $M = 64$, $P \approx 39,23$.
19. (5 points) $B = 512$, $M = 128$, $P \approx 15,00$.
20. (5 points) $B = 512$, $M = 256$, $P \approx 4,00$.

Zusätzlich erfüllen alle Tests $N \cdot M \leq 120.000$, wobei N die maximale Zahl an Schreiboperationen ist, die das Programm ausführen soll.

Während des Wettbewerbs wird die Lösung für jede Gruppe auf einer kleinen Testmenge ausgeführt. Nachdem der Wettbewerb beendet ist, wird die letzte Einsendung und die Einsendung mit der höchsten Punktezahl auf einer größeren Testmenge ausgeführt, und das beste Ergebnis daraus ist die finale Punktezahl für diese Aufgabe. **Die Punktezahl, die du während des Contests im CMS siehst ist also nicht deine endgültige Punktezahl!** Dies wird gemacht, um die Genauigkeit deiner erreichten Punktezahl zu erhöhen. Das Bewerten auf dem ganzen Test wäre zu langsam, um es bei jeder Submission auszuführen.