Pamięć flash

1 s. / 10 s. **1 GB**

Mikrokontroler ma wbudowaną pamięć flash o B bitach. Będziemy musieli w niej przechowywać i aktualizować M-bitowe wartości. Pamięć flash ma jednak pewne ograniczenie. O ile wartości bitów można zmieniać z `0` na `1` bez żadnych problemów, to aby wyzerować którykolwiek bit (zmienić z `1` na `0`), musimy wyczyścić całą pamięć. Pamięć może być wyczyszczona jedynie ograniczoną liczbę razy, po czym się zużywa i mikrokontroler musi być wymieniony na nowy. Z tego powodu przed wyczyszczeniem pamięci chcielibyśmy zapisać w niej możliwie najwięcej wartości.

Twoim zadaniem jest zaprojektowanie efektywnego sposobu przechowywania wartości w pamięci mikrokontrolera w taki sposób, żeby zawsze dało się odczytać obecnie przechowywaną wartość. Innymi słowy, Twój program powinien zawierać implementację dwóch operacji:

- Zapis wartości: Na wejściu będzie znajdować się obecny stan pamięci oraz wartość do zapisania. Twój program powinien wypisać stan pamięci po zapisie.
- Odczyt wartości: Wejście zawiera stan pamięci po pewnej liczbie operacji zapisu. Twój program powinien wypisać wartość zapisaną w ostatniej operacji.

Twoje operacje czytania z pamięci oraz zapisywania do pamięci nie mogą wymieniać informacji inaczej niż poprzez czytanie obecnego stanu pamięci podanego na wejściu oraz operację zapisywania do pamięci poprzez zmianę pewnej liczby bitów (być może zeru) z `0` na `1` i wypisanie tak uzyskanego stanu pamięci na wyjście.

Interakcja. To zadanie jest interaktywne. Na początku działania programu w pierwszym wierszu standardowego wejścia będzie dana liczba całkowita T. $T = 0$ oznacza, że zadaniem Twojego programu będzie zapisywanie wartości do pamięci, a w przypadku $T = 1$ odczytywanie wartości z pamięci. Drugi wiersz wejścia zawiera dwie liczby całkowite B i M. W kolejnych wierszach będą znajdowały się opisy kolejnych operacji.

W przypadku obu operacji pierwszy wiersz opisu zawiera liczbę całkowitą C. Jeśli $C = 0$, Twój program powinien natychmiast zakończyć działanie. W przeciwnym wypadku ($C = 1$) w kolejnych wierszach następuje opis operacji, jak poniżej:

- Dla $T = 0$ i $C = 1$, drugi wiersz opisu zawiera dwa wyrazy oddzielone spacją: obecny stan pamięci przedstawiony jako ciąg B bitów oraz wartość do zapisania w pamięci, przedstawiona jako ciąg M bitów. Jeśli Twój program jest w stanie zapisać wartość do pamięci jedynie zmieniając bity z `0` na `1`, powinien wypisać liczbę 1 oraz w kolejnym wierszu nowy stan pamięci w postaci ciągu B bitów. Jeśli Twój program nie jest w stanie zapisać nowej wartości do pamięci, powinien wypisać pojedynczy wiersz zawierającą liczbę 0.
- Dla $T = 1$ i $C = 1$, drugi wiersz opisu zawiera jeden wyraz: stan pamięci jako ciąg B bitów. Twój program powinien wypisać pojedynczy wyraz: ciąg M bitów oznaczających wartość, która była zapisana w pamięci jako ostatnia.

Przykład.

Wejście	Wyjście
`0`	
`6 2`	
`1`	
`111111 00`	
	`0`
`1`	
`000000 11`	
	`1`
	`110000`
`0`	

W tym przykładzie program będzie zapisywał 2-bitowe wartości do 6-bitowej pamięci. Pierwsze zapytanie to zapis wartości `00`, czego program nie był w stanie zrobić. Drugie zapytanie to zapis wartości `11`, co już mu się udało. Zwróć uwagę, że stan pamięci w drugim zapytaniu jest niezależny od stanu wypisanego po pierwszym zapytaniu.

Przykład.

Wejście	Wyjście
`1`	
`6 2`	
`1`	
`110000`	
	`11`
`1`	
`110100`	
	`01`
`0`	

W tym przykładzie program będzie czytał 2-bitowe wartości z 6-bitowej pamięci. W pierwszym zapytaniu stan pamięci to `110000`, z którego program odczytał wartość `11`. W pierwszym zapytaniu stan pamięci to `110100`, z którego program odczytał wartość `01`.

Uwaga. Aby zagwarantować, że Twoje odpowiedzi trafiają do systemu oceniającego, musisz czyścić bufor strumienia wyjścia po każdej odpowiedzi (zauważ także, że wyjście zawsze kończy się znakiem nowego wiersza):

Język	Komenda
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()`

Testowanie. System sprawdzający dla każdego testu uruchomi cztery instancje Twojego programu jednocześnie, dwie zapisujące oraz dwie odczytujące. Limit czasu oraz pamięci dotyczy czasu i pamięci zużytych przez wszystkie te instancje łącznie. **Każda celowa próba przekazania danych poza wejściem/wyjściem będzie uznana za oszustwo i może być przyczyną dyskwalifikacji.**

Pewna liczba bloków pamięci, każda zawierająca B bitów, zostanie zainicjalizowana zerami.

Następnie będą wykonywane na nich operacje zapisywania i odczytywania wartości w pewnej poprawnej kolejności.

Na początku operacji zapisywania, instancja zapisująca dostanie aktualny stan jednego z bloków oraz wartość, którą ma zapisać. Możesz założyć, że wartości do zapisania zostały wybrane losowo i niezależnie z jednakowym prawdopodobieństwem z przedziału $0 \dots 2^M - 1$. Jeżeli Twój program potrafił zapisać wartość do pamięci, stan bloku zmienia się na ten zwrócony przez Twój program. W przeciwnym wypadku, blok nie zostanie już użyty przez żadne przyszłe operacje zapisywania.

Podczas operacji odczytywania, instancja odczytująca dostanie stan bloku po poprawnej operacji zapisywania. Następnie sprawdzane jest czy zwrócona wartość jest taka sama jak wartość, która miała być zapisana podczas operacji zapisywania. Operacje odczytywania są wywołane dokładnie raz dla każdego wyjścia poprawnie wykonanej operacji zapisywania.

Ocenianie. Dla każdej grupy testów, wynik Twojego programu będzie proporcjonalny do średniej liczby wartości zapisanych w testach należących do tej grupy. Dokładniej, jeśli Twój program zapisał w bloku średnio V wartości, to otrzyma $100 \cdot V/P\%$ punktów za daną grupę. Wartości P są wylistowane poniżej. Jeżeli Twój program niepoprawnie odczyta dowolną wartość, wynikiem dla całej grupy będzie zero. Dla każdego innego błędu, liczba wartości poprawnie przeczytana w danym teście będzie liczona jako zero.

Grupy testów spełniają następujące warunki:

1. (5 punktów) $B = 16$, $M = 8$, $P = 4.062445024495069624056$.
2. (5 punktów) $B = 32$, $M = 8$, $P = 12.264904841300964834177$.
3. (5 punktów) $B = 32$, $M = 16$, $P = 4.129591513707784802006$.
4. (5 punktów) $B = 64$, $M = 8$, $P = 30.039277894268828900030$.
5. (5 punktów) $B = 64$, $M = 16$, $P = 12.953148094217360432715$.
6. (5 punktów) $B = 64$, $M = 32$, $P = 4.073559788233661501537$.
7. (5 punktów) $B = 128$, $M = 8$, $P = 69.777892228928747548775$.
8. (5 punktów) $B = 128$, $M = 16$, $P = 34.7317912751436352409 76$.
9. (5 punktów) $B = 128$, $M = 32$, $P = 13.950788987705638908663$.
10. (5 punktów) $B = 128$, $M = 64$, $P = 4.039918210604800133907$.
11. (5 punktów) $B = 256$, $M = 8$, $P = 174.468047086071038511453$.
12. (5 punktów) $B = 256$, $M = 16$, $P = 82.2226141514041773345 54$.
13. (5 punktów) $B = 256$, $M = 32$, $P = 37.629382269769206488916$.
14. (5 punktów) $B = 256$, $M = 64$, $P = 14.263462282054140577686$.
15. (5 punktów) $B = 256$, $M = 128$, $P = 4.015569093893943430859$.
16. (5 punktów) $B = 512$, $M = 16$, $P = 204.7462421274103461702 21$.
17. (5 punktów) $B = 512$, $M = 32$, $P = 91.778595148073111539847$.
18. (5 punktów) $B = 512$, $M = 64$, $P = 39.230279242145938712621$.
19. (5 punktów) $B = 512$, $M = 128$, $P = 15.000000002167672268601$.
20. (5 punktów) $B = 512$, $M = 256$, $P = 4.005423277111055468876$.

Dodatkowo, wszystkie testy spełniają $N \cdot B \leq 120\,000$, gdzie N to maksymalna liczba operacji zapisywania, o których wykonanie poproszony będzie Twój program.

W trakcie zawodów Twoje rozwiązanie będzie ocenione jedynie na części testów z każdej grupy. Po zakończeniu zawodów, ostatnie zgłoszenie oraz zgłoszenie, które otrzyma najwięcej punktów z testów częściowych, zostanie ponowne ocenione na pełnych testach. Lepszy z wyników uzyskanych przez te dwa programy będzie Twoim ostatecznym wynikiem za to zadanie. **Wynik widoczny w CMS w trakcie zawodów nie jest ostateczny.** Celem tego zabiegu ma być zwiększenie dokładności Twojego wyniku. Ocenianie na pełnych testach w trakcie zawodów mogłoby okazać się za wolne.