Флеш-память

1 сек / 10 сек **1 GB**

Микроконтроллер имеет встроенную флеш память, состоящую из B-битовых ячеек. Мы будем несколько раз записывать в одну такую ячейку различные M-битовые значения. У флеш-памяти есть ограничение: отдельные биты можно свободно менять с `0` на `1`, но менять их обратно с `1` на `0` возможно только при стирании всей ячейки сразу. Стирать ячейку можно лишь ограниченное количество раз, после чего чип изнашивается и требует замены. Поэтому желательно уметь записывать в ячейку новые значения как можно больше раз, не прибегая к стиранию.

Ваша задача – придумать эффективный способ записи значений в ячейку флеш-память таким образом, чтобы последнее записанное значение всегда можно было бы считать. Конкретнее, вам необходимо написать программу, реализующую две операции:

- Запись значения: На вход операции даётся изначальное состояние ячейки памяти и новое значение, которое необходимо записать. Необходимо вывести новое состояние ячейки.
- Считывание значения: Входом данной операции является состояние ячейки памяти после некоторого количества операций записи. Необходимо вывести последнее записанное в ячейку значение.

Операции записи и чтения не могут обмениваться никакой информацией помимо считывания изначального состояния ячейки памяти, изменения некоторого (возможно нулевого) числа бит с `0` на `1`, и выдачи нового состояния.

Общение. Это интерактивная задача. После запуска программы, первая строка ввода содержит целое число T, где $T = 0$ означает, что программа будет записывать значения в память, а $T = 1$ означает, что она будет считывать значения. На второй строке даны целые числа B и M. Следующие строки описывают операции.

Как для операций записи, так и для чтения, каждая операция начинается со строки, на которой дано целое число C, где $C = 0$ означает, что больше запросов нет и программа должна завершить работу, тогда как $C = 1$ означает, что программа должна продолжать.

- Если $T = 0$ и $C = 1$, следующая строка содержит две разделённых пробелом двоичных строки: изначальное состояние памяти в виде последовательности длиной B бит, и новое значение, которое необходимо записать, в виде последовательности длиной M бит. Если ваша программа способна записать новое значение, поменяв некоторое количество бит с `0` на `1`, она должна сначала вывести число 1, а за ним, на следующей строке, новое состояние памяти в виде последовательности из B бит. Если программа не способна записать новое значение в память, она должна вывести число 0.
- Если $T = 1$ и $C = 1$, на следующей строке дано состояние памяти в виде последовательности B бит, а ваша программа должна вывести последовательность M бит – последнее записанное в память значение.

Пример. Входные данные / Выходные данные

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

В этом примере программа запущена в режиме записи 2-битных значений в 6-битную память. Первый запрос требует записать значение `00`, что программа осуществить не способна. Второй запрос требует записать значение `11`, что программа может осуществить. Обратите внимание, что состояние памяти во втором запросе не является выводом вашей программы после первого запроса.

Пример. Входные данные / Выходные данные

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

В этом примере программа запущена в режиме считывания 2-битных значений из 6-битной памяти. В первом запросе необходимо считать значение из памяти в состоянии `110000`, и программа распознаёт значение `11`. Во втором запросе требуется считать значение из памяти `110100`. Здесь программа считывает результат `01`.

Примечание. Для того, чтобы ваши выводы были правильно зачитаны оценивающей системой, необходимо сбрасывать буфер после каждого ответа (также заметьте, что вывод всегда завершается переносом строки):

Язык	Код
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()`

Тестирование. Для каждого теста будут одновременно запущены 4 копии вашей программы, 2 из них в режиме записи, а 2 – в режиме чтения. Ограничения по процессорному времени и памяти применяются для всех этих четырёх копий вместе взятых. **Любая целенаправленная попытка передавать данные между программами какими-либо, не предусмотренными задачей способами, будет считаться жульничеством и может привести к дисквалификации.**

Несколько ячеек памяти размером B бит будут инициализированы нулями. Следующие операции будут применяться в произвольном разрешённом условиями порядке пока это возможно.

Во время операции записи, экземпляру программы, запущенному в режиме записи, будет дано состояние одной из ячеек, а также новое значение, которое необходимо туда записать. Можно предполагать, что значения для записи будут выбираться независимо и случайно из промежутка $0 \dots 2^M - 1$. Если ваша программа способна записать значение, состояние соответствующей ячейки будет обновлено на то, что выдала ваша программа. Если ваша программма не способна записать значение, ячейка больше не будет использоваться для записи.

Во время операции считывания, программе, запущенной в режиме считывания, будет дано состояние одной из ячеек после успешной записи. Оценивающая система проверяет, чтобы считанное вашей программой значение соответствовало тому, что было последним записано в ячейку. Операция считывания будет применена к выводу каждой успешной операции записи ровно один раз.

Оценивание. В каждой группе тестов, оценка программы пропорциональна среднему числу значений, которое программа смогла записать (а затем считать) в одну ячейку в этом тесте. Точнее, если вашей программе удаётся записать в ячейку в среднем V значений, она получит $100 \cdot V/P\%$ от максимума очков за соответствующую группу тестов, где P – параметр, данный ниже. Если ваша программа возвращает неверное значение в любой операции чтения, вся группа тестов получит 0 очков. Если программа некорректно завершает работу или ошибается каким-либо другим образом в одном из тестов, количество верно записанных (и прочитанных) значений в том тесте считается равным нулю.

Группы тестов удовлетворяют следующим условиям:

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

18. (5 очков) $B = 512$, $M = 64$, $P = 39.230279242145938712621$.
19. (5 очков) $B = 512$, $M = 128$, $P = 15.000000002167672268601$.
20. (5 очков) $B = 512$, $M = 256$, $P = 4.005423277111055468876$.

В дополнение, все тесты удовлетворяют условию $N \cdot B \leqslant 120\,000$, где N – максимальное количество операций записи, которое может понадобиться совершить за весь тест.

Во время соревнования ваши решения будут оценены на предварительном, частичном наборе тестов. После соревнования, ваше последнее посланное решение, а также решение, набравшее максимальное количество очков на предварительном тестовом наборе, будут протестированы на полноценном, более крупном наборе данных, и в финальный зачёт пойдёт лучший из полученных двумя решениями на этом наборе результат. **Очки, которые вы видите в системе CMS не являются вашим окончательным результатом.** Цель такого оценивания – повысить точность оценки результата, так как оценивание на полном тестовом наборе во время соревнования было бы слишком медленным.