### Замечание

Во всех задачах, если не указано особое, программа должна производить ввод со стандартного потока ввода или файл input.txt, выводить на стандартный поток или файл output.txt, ограниеение по времени 2 с, ограничение по памяти 256MB.

## Задача A. Цифровой корень

Цифровой корень натурального числа — сумма цифр от суммы цифр от суммы цифр ... числа. Например, для числа 345 цифровой корень 3 (345 -> 12 -> 3)

Ваша задача для заданного цифрового корня $K$ найти $N$-значное число, цифровой корень которого равен $K$ и в котором нет подряд двух одинаковых цифр

### Формат входных данных

Вводится два целых числа $N$ и $K$ ($1 \leqslant N \leqslant 10^5, 0 \leqslant K \leqslant 9$)

### Формат выходных данных

Требуется вывести искомое число без ведущих нулей или -1, если такого числа не существует

### Пример

| стандартный ввод | стандартный вывод |
|---|---|
| 3 3 | 345 |

## Задача B. Перфекционист Глеб

Холодным мартовским утром Глеб играл на улице в игру: для начала он придумывает перестановку из чисел от 1 до $N$ — $a$, также запоминая $a_R$ - перестановку получающаяся из $a$ разворотом. Потом он выбирает число $K$, и выписывает $K$ раз подряд последовательность, полученную склеиванием $a$ и $a_R$. Глеб очень любит чтобы у полученной последовательности длина наибольшей возрастающей подпоследовательности равнялась $N$. Помогите Глеб найти такое минимальное $K$.

### Формат входных данных

Первая строка входных данных содержит число $N$ ($1 \leqslant N \leqslant 300\,000$)

Во второй строке записаны $N$ целых чисел $a_i$ ($1 \leqslant a_i \leqslant n$, $a_i \neq a_j$ для $i \neq j$) — загаданная перестановка.

### Формат выходных данных

В единственной строке выведите искомое $K$. Гарантируется, что такое $K$ существует.

### Примеры

| стандартный ввод | стандартный вывод |
|---|---|
| 3<br>3 2 1 | 1 |
| 3<br>2 3 1 | 2 |
| 4<br>3 4 1 2 | 2 |
| 4<br>4 3 1 2 | 1 |

## Задача C. Связность графа

Однажды на день рождение Ивану подарили граф состоящий из $N$ вершин в котором вершины с номерами $x$ и $y$ соединены ребром тогда и только тогда, когда $gcd(x, y) \geqslant g$, где

### Формат входных данных

Первая строка входных данных содержит три целых числа $N$, $g$, $M$ и ($1 \leqslant N, g, M \leqslant 200\,000$) — количество вершин в графе и параметр графа и количество запросов на связность.

Далее следуют $m$ строк, описывающих запросы, каждый запрос задан двумя числами $x$ и $y$, ($1 \leqslant x, y \leqslant N, x \neq y$).

### Формат выходных данных

Выведите $M$, по одной на каждый запрос, соответственно «Yes» , если вершины находятся в одной компоненте, и «No» иначе.

### Пример

| стандартный ввод | стандартный вывод |
|---|---|
| 6 2 4<br>2 3<br>4 3<br>5 1<br>2 6 | Yes<br>Yes<br>No<br>Yes |

## Задача D. ЧПУ-фрезер

В этой задаче вам требуется написать программу для фрезерного станка с ЧПУ по изображению заготовки. Изображение задается в виде прямоугольника $N \cdot M$,

где $N$ – количество строк, а $M$ – количество столбцов. Символ '.' означает, что эта клетка не должна быть обработана. Символ '*' означает, что эта клетка должна быть отфрезерована.

Фреза станка изначально находиться в точке с координатой $(0, 0)$ (левый верхний угол картинки) и может за одну секунду переместиться в одну из соседних клеток (команды `N` - вверх, `E` - вправо, `S` - вниз , `W` - влево). Кроме того, в каждый момент времени фреза может находиться или в опущенном состоянии и тогда происходит процесс обработки, или в поднятом состоянии и тогда обработка не происходит. Подъем и опускание фрезы занимает 1 секунду. Обработка клетки происходит за одну секунду.

В процессе обработки заготовки происходит нагрев фрезы в опущенном состоянии на 1 градус за одну секунду и охлаждение на 1 градус в поднятом состоянии. Соответственно, команда D позволяет опустить фрезу и команда U позволяет поднять фрезу. Будем считать, что время опускания и поднятия фрезы пренебрежимо малы.

Начальная и минимальная температура 30 градусов. При нагреве больше 50 градусов фреза ломается. Если фреза выходит за границы изображения, то она ломается.

Ваша задача сгенерировать программу для фрезера из команд `N, E, S, W, U, D`, которая получает заданное изображение.

Программа для фрезера оценивается по времени получения изображения. Программы которые приводят к поломке фрезы считаются неверными.

### Формат входных данных

Изображение описывается числами $N$ и $M$ ($1 \leqslant N, M \leqslant 1000$), затем идет $N$ строк по $M$ символов '.' или '*'

### Формат выходных данных

Вывести требуется программу из команд 'N','E','S','W','U','D' в одну строку без пробела, которую должен выполнить станок для получения изображения. Длина программы фрезера не должна превышать $10^6$ символов.

### Система оценки

Итоговый балл за каждый тест и подзадачу будет выставляться в зависимости от лучшего полученного результата среди участников после олимпиады.

### Пример

| стандартный ввод | стандартный вывод |
|---|---|
| 5 5<br>....*<br>....*<br>*...*<br>....*<br>***** | EEEEDSUWWWWSDUEEEEDSUWWWWSDEEEE |

### Замечание

В задаче есть несколько подзадач с файлам изображений, для которых требуется сгенерировать и отправить на проверку файл с программой. Отдельной подзадачей требуется отправить программу на одном из языков программирования, которая для произвольного изображения получает программу для станка.

При проверке файлов с изображениями будет доступен протокол проверки корректности программы для станка и время работы для программы не приводящей к поломке.

Подзадача со сдачей программы на языке программирования будет проверяться на всех тестовых изображениях во время олимпиады и на некотором наборе тестов после окончания олимпиады.

## Задача E. RW

Многопоточность — свойство платформы (например, операционной системы, виртуальной машины и т. д.) или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени.

В этой задаче требуется написать программу, которая корректно работает многопоточно. Есть область общей памяти, позволяющая чтение и запись. Несколько параллельно работающих функций имеют к ней доступ, при этом одновременно могут читать сколько угодно потоков, но писать — только один. Необходимо обеспечить корректный доступ к этой памяти.

В этой задаче требуется написать программу на псевдоязыке для функции читателей области памяти и писателей в эту область. Так как параллельное чтение и запись в общую область памяти может привести к некорректному поведения, требуется избежать случая, когда несколько пишущих функций могут одновременно писать и случая когда читатель читает, а писатель пишет одновременно.

Реализации такого ограничения могут быть различными, но нас интересует случай приоритета писателей. Как только появился хоть один писатель, читателей

больше не пускать. При этом читатели могут простаивать.

Для того чтобы обеспечить логику ожидания существует специальная общая переменная-*mutex* для которой есть две операции *enter* и *leave*. Операция *enter* успешно выполняется если никакая другая функция не выполнила с этой переменной операцию *enter* не выполнив операцию *leave*. В противном случае выполнения функции останавливается до тех пор, пока та функция, которая сделала операцию *enter* не выполнит операцию *leave*. Если на операции *enter* ожидает несколько функций, то успешно выполнить её может только одна.

Вы можете завести не более 10 глобальных переменных-*mutex* и не более 10 целочисленных переменных изначально равных нулю:

```
global
  int x;
  int y;
  mutex z;
```

где `int x` - объявление целочисленной переменной `x`, `mutex z` - объявление переменной-*mutex* `x`. Имя переменной может состоять из одной латинской строчной буквы. Имена переменных не могу повторяться.

Прототип функции читателя выглядит так:

```
reader_begin
//  тут можно добавить свой код вместо комментария
read()
// тут можно добавить свой код вместо комментария
reader_end
```

Прототип функции писателя выглядит так:

```
writer_begin
// тут можно добавить свой код вместо комментария
write()
// тут можно добавить свой код вместо комментария
writer_end
```

В качестве операций можно использовать следующие конструкции:

- `z.enter();` - операция `enter` для переменной-`mutex` z
- `z.exit();`
- `x.increase();` - увеличить целочисленную переменную `x` на 1
- `x.decrease();` - уменьшить целочисленную переменную `x` на 1
- `if (x==<константа>) <другая операция>` выполнение операции при выполнении условия

При выполнении read() и write() и происходят соответствующие чтения и записи общей памяти.

### Формат выходных данных

Необходимо сдать программу на этом псевдоязыке, которая решает задачу. Например,

```
global
  int x;
  mutex y;

reader_begin
  x.increase();
  read();
  if (x == 1) x.decrease();
reader_end

writer_begin
  y.enter();
  write();
  y.leave();
writer_end
```

В программе должны остаться вызовы read() и write()

### Замечание

Программа из примера не является правильным решением и приведена только для иллюстрации формата. Считайте что функции читателей и писателей могут работать параллельно в нескольких экземплярах

## Задача F. Квадрокоптеры

Однажды Федор решил провести чемпионат мира по программированию. Ясно что такое соревнование нельзя провести без торжественного открытия, а открытие без красочного шоу. К счастью у Федора есть $N$ квадрокоптеров и он решил построить из них заданную фигуру в зале над участниками.

Управление квадрокоптера осуществляется следующим образом. Изначально в квадрокоптер загружается программа, которая каждый момент времени решает куда должен переместиться квадрокоптер - влево, вправо, вверх, вниз или остаться на месте (эти команды кодируются числами от 1 до 5 соответственно). Каждый момент времени каждый квадрокоптер принимает решение о перемещении. После этого все квадрокоптеры одномоментно совершают это перемещение.

У каждого квадрокоптера есть 10 байт памяти которую он может изменять каждый момент времени. У каждого квадрокоптера есть сенсоры которые позволяют считать состояние памяти соседних (по сторонам) квадрокоптеров или квадрокоптеров в той же клетке.

Представим зал в виде прямоугольной сетки, к некоторых клетках которой находятся квадрокоптеры. Изначально они находятся в неизвестных различных координатах, но гарантируется что начальная конфигурация является связной по сторонам фигурой.

Вам требуется написать программу которая выполняется на квадрокоптере.

При формировании фигуры квадрокоптерами важно только относительное положение квадрокоптеров относительно друг-друга. То есть требуется чтобы в какой-то момент времени относительная конфигурация должна совпадать с заданной.

## Формат входных данных

Каждый момент времени на вход программе подается неизменяемая целевая конфигурация квадрокоптеров: число $N$ ($1 \leqslant N \leqslant 20$) и $N$ пар целых положительных чисел (каждое из которых до 100) - точки в которых могут находиться квадрокоптеры, чтобы сформировать нужную конфигурацию.

Следующие 10 чисел задают состояние памяти текущего квадрокоптера, на котором выполняется программа - числа от 0 до 255, разделенные проблемами

Затем следует число соседних квадрокоптеров $M$. Следующие $M$ строк задают состояние каждого из квадрокоптеров - первое число от 1 до 5 где он находится (слева, справа, сверху, снизу, в той же клетке) и 10 чисел от 0 до 255 - состояние его памяти.

## Формат выходных данных

Требуется вывести текущее действие квадрокоптера - число от 1 до 5, затем 10 байт - новое состояние памяти

## Замечание

Для каждого хода для каждого квадрокоптера запускается решение с данными, описанными во входном формате. Программа делающая один ход не должна предполагать что можно как-то сохранить данные кроме как в 10 байтах памяти квадрокоптера.

Пример ввода для одного хода:

```
4
0 0
1 1
2 2
3 3
0 1 200 0 0 0 0 0 0 0
3
1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0 0
```

Пример вывода:

```
2
0 1 199 0 0 0 0 0 0 0
```