D. Garden Decorations

Problem Name	Garden Decorations
Time Limit	7 seconds
Memory Limit	1 gigabyte

Каждый день, идя в школу и возвращаясь домой, Детье проходит по улице с N домами, пронумерованными от 0 до $N-1$. В настоящее время в доме i живет человек i. Чтобы сменить обстановку, жители решили поменяться домами друг с другом. В дом i переедет человек a_i (который сейчас живет в доме a_i).

В саду каждого дома есть статуя птицы. У статуй есть два возможных состояния: либо крылья *раскрыты* (как будто птица летит), либо *закрыты* (как будто она стоит на земле). У жителей дома очень сильные предпочтения относительно того, как должны выглядеть их статуи птиц: они отказываются переезжать в новый дом, пока статуя в новом саду не будет выглядеть так же, как в предыдущем. Детье хочет помочь им исправить статуи птиц так, чтобы они могли переехать.

Для этого она делает следующее: каждый раз, проходя по улице (по дороге в школу или домой), она наблюдает за статуями птиц, которые проходит одну за другой, и, возможно, меняет некоторые статуи (раскрывая или закрывая их крылья). Поскольку ее дни в школе и дома очень насыщены, **она не помнит, в каком состоянии были птицы, которых она видела во время предыдущих прогулок**. К счастью, она записала список $a_0, a_1, ..., a_{N-1}$, поэтому она всегда знает, какой житель куда планирует переехать.

Помогите Детье разработать стратегию, которая подскажет ей, каких птиц нужно менять, чтобы привести статуи в соответствие с пожеланиями жителей. Она может пройти по улице не более 60 раз, но для получения более высокого результата ей следует пройти по улице меньшее число раз.

Implementation

В этой задаче ваше решение будет запущено на каждом тесте несколько раз.

В каждом запуске вы должны сначала прочитать строку с двумя целыми числами w и N — индексом прогулки и количеством домов соответственно. В первом запуске вашей

программы $w = 0$, во втором $w = 1$, и так далее (подробнее описано ниже).

Во второй строке ввода находится N целых чисел $a_0, a_1, ..., a_{N-1}$, означающих, что человек, который будет переезжать в дом i, в настоящее время живет в доме a_i. Числа a_i образуют *перестановку*: то есть каждое число от 0 до $N - 1$ встречается в списке a_i ровно один раз. Обратите внимание, что житель может никуда не переезжать, то есть $a_i = i$.

Жители меняются домами только один раз. Это означает, что для фиксированного теста значение N и список a_i будут одинаковыми для всех запусков вашей программы.

First Run.

Для первого запуска вашей программы $w = 0$. В этом запуске вы должны просто вывести одно целое число W ($0 \le W \le 60$) — количество раз, которое вы хотите, чтобы Детье прошла мимо домов. Затем ваша программа должна завершиться. После этого ваша программа будет запущена еще W раз.

Subsequent Runs.

В следующем запуске вашей программы $w = 1$, в следующем $w = 2$ и так далее до последнего запуска, в котором $w = W$.

После того, как вы прочитали w, N и значения $a_0, a_1, ..., a_{N-1}$, Детье начинает идти по улице.

- Если w нечетное, то Детье идет из дома в школу, и она будет проходить мимо домов в порядке $0, 1, ..., N - 1$.

  Теперь ваша программа должна прочитать строку, состоящую из единственного символа b_0, либо 0 (закрыто), либо 1 (раскрыто) — текущее состояние статуи перед домом 0. После считывания b_0 вы должны вывести строку, состоящую из единственного символа 0 или 1 — новым значением, которое вы хотите установить для b_0.

  Затем ваша программа должна прочитать строку, состоящую из единственного символа b_1 — состоянием статуи перед домом 1; и вывести новое значение b_1. Так продолжается для каждого из N домов. После прохождения последнего дома (т.е. чтения и записи b_{N-1}) **ваша программа должна завершиться**.

  Обратите внимание, что ваша программа сможет прочитать следующее значение b_{i+1} только после того, как вы запишете значение b_i.

- Если w четное, то Детье идет из школы домой пешком и проходит мимо домов в обратном порядке $N - 1, N - 2, ..., 0$.

Процесс такой же, как и при нечетном w, за исключением того, что начинается с чтения и записи b_{N-1}, затем b_{N-2}, и так далее до b_0.

Если $w = 1$, входные значения $b_0, b_1, ..., b_{N-1}$ — это исходное состояние статуй птиц. Если $w > 1$, входные значения $b_0, b_1, ..., b_{N-1}$ для вашей программы будут такими, какими их установил предыдущий запуск вашей программы.

В конце, после финального запуска вашей программы, значение b_i должно быть равно исходному значению b_{a_i} для всех i, иначе вы получите вердикт Wrong Answer.

Details.

Если *сумма* времен выполнения $W + 1$ отдельных запусков вашей программы превысит лимит времени, вы получите вердикт Time Limit Exceeded.

Убедитесь, что вы сбрасываете буффер после печати каждой строки, иначе ваша программа может получить вердикт Time Limit Exceeded. В Python это происходит автоматически, если вы используете `input()` для чтения строк. В C++, `cout << endl;` сбрасывает буффер в дополнение к печати новой строки; если вы используется printf, используйте `fflush(stdout)`.

Constraints and Scoring

- $2 \leq N \leq 500$.
- Вы можете использовать не более $W \leq 60$ запусков.

Ваше решение будет протестировано на наборе подзадач, каждая из которых даёт определенное количество очков. Каждая подзадача содержит набор из нескольких тестов. Чтобы получить очки за подзадачу, вам нужно решить все тесты в этой подзадаче.

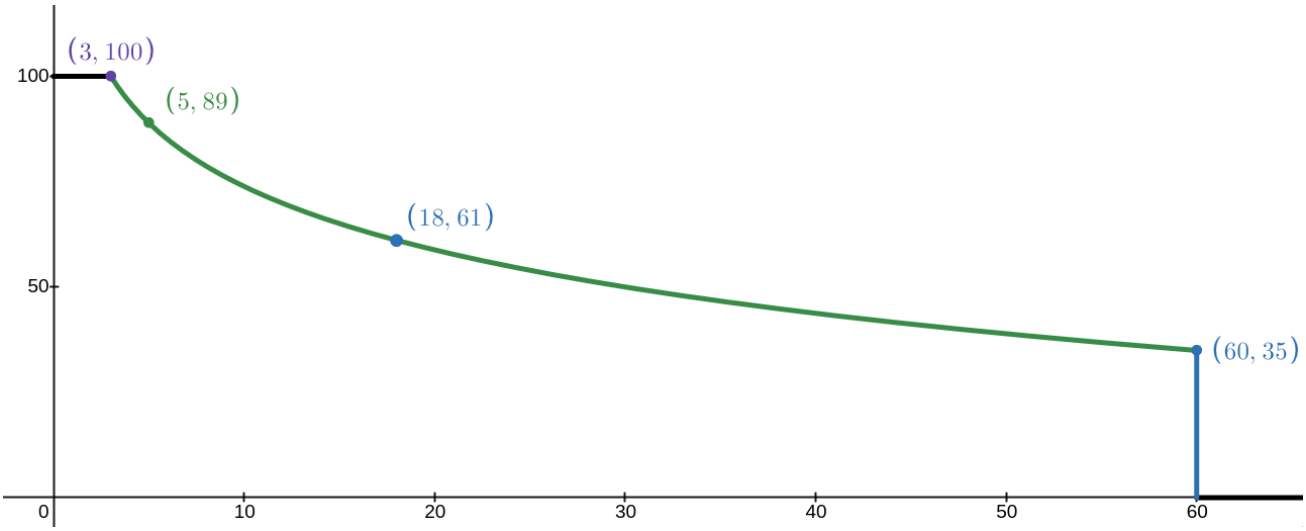
Подзадача	Максимальный балл	Ограничения
1	10	$N = 2$
2	24	$N \leq 15$
3	9	$a_i = N - 1 - i$
4	13	$a_i = (i + 1) \bmod N$
5	13	$a_i = (i - 1) \bmod N$
6	31	Без дополнительных ограничений

За каждую подзадачу, которую ваша программа решает правильно, вы получаете оценку по следующей формуле:

$$\text{score} = S_g \cdot \left(1 - \tfrac{1}{2}\log_{10}(\max(W_g, 3)/3)\right),$$

где S_g — максимальный балл для подзадачи, а W_g — максимальное значение W используемое в этой подзадаче. Ваш результат для каждой подзадачи будет округлен до ближайшего целого числа.

На графике ниже показано количество очков в зависимости от W, которое получит ваша программа, если решит все подзадачи с одинаковым значением W. В частности, чтобы набрать 100 баллов по этой задаче, нужно решить каждый тест с $W \leq 3$.



Testing Tool

Чтобы облегчить тестирование вашего решения, мы предлагаем простую программу, которую вы можете скачать. См. раздел «Attachments» внизу страницы с задачей в системе Kattis. Инструмент можно использовать по желанию. Обратите внимание, что официальная программа для тестирования на Kattis отличается от предложенной программы.

Чтобы воспользоваться программой, создайте входной файл, например «sample1.in», который должен начинаться с числа N, затем идет строка с N числами, задающими перестановку, и еще одна строка с N числами (0 или 1), задающими начальные состояния птиц. Например:

```
6
1 2 0 4 3 5
1 1 0 0 1 0
```

Для решения на Python, например `solution.py` (обычно запускаются с `pypy3 solution.py`):

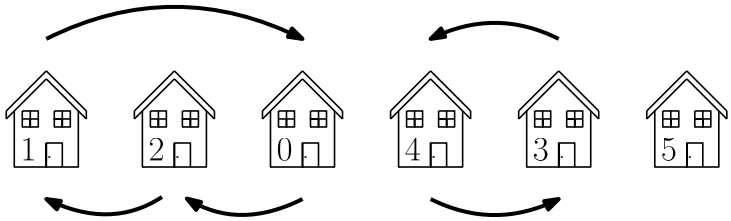
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Для решения на C++ сначала скомпилируйте их (например с `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) и затем запустите:

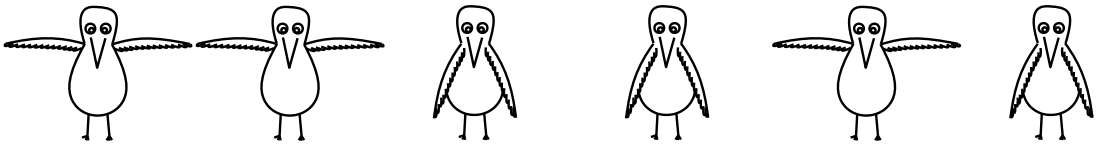
```
python3 testing_tool.py ./solution.out < sample1.in
```

Example

В примере дана следующая перестановка людей в домах:

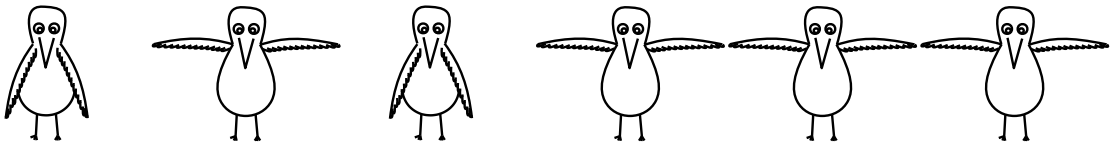


При первом запуске программы-примера (с $w = 0$) она выдает $W = 2$, что означает, что Детье пройдет по улице два раза (и программа будет запущена ещё два раза). Перед первой прогулкой птицы в садах выглядят следующим образом:



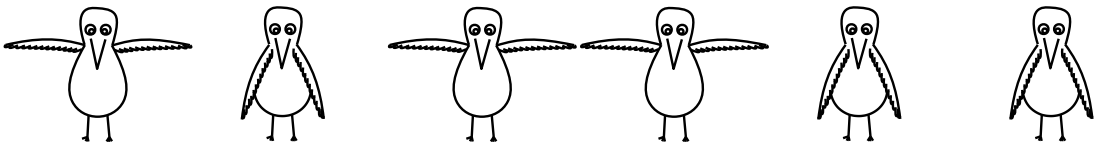
Затем программа запускается с $w = 1$: это означает первую прогулку Детье. Она проходит птиц одну за другой, начиная слева, и, возможно, меняет их состояние. Программы-пример должена вывести состояние i-ой птицы до того, как мы увидим $(i + 1)$-ую птицу.

После того как Детье пришла в школу, состояние птиц выглядит следующим образом:



В последнем запуске программы (с $w = 2$), Детье возвращается из школы домой. Помните, что в этом случае она будет проходить мимо птиц справа налево и обрабатывать их в обратном порядке! Это означает, что ей нужно определить состояние i-ой птицы, прежде чем она увидит $(i - 1)$-ую.

После того как она закончит свою прогулку, птицы будут выглядеть следующим образом:



Действительно, это правильная конфигурация. Например, статуя птицы 3 (т. е. четвертая слева) раскрыта (теперь $b_3 = 1$), что верно, так как человек 4 будет перемещаться туда ($a_3 = 4$), а в его саду статуя птицы изначально была раскрыта (изначально $b_4 = 1$).

grader output	your output
0 6	
1 2 0 4 3 5	
	2

grader output	your output
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

grader output	your output
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1