Permutation

Фараоны используют взаимное расположение и гравитацию планет для ускорения собственных космических кораблей. Предположим, что корабль собирается совершить полет вблизи n планет с орбитальными скоростями $p[0], p[1], \ldots, p[n-1]$, ровно в таком порядке. Для каждой из посещаемых планет ученые Фараонов могут выбрать, хотят ли они использовать эту планету для ускорения. Однако, для сохранения энергии, ученые соблюдают следующее правило: после ускорения с помощью планеты с орбитальной скоростью $p[i]$, корабль не может быть ускорен с использованием планеты с орбитальной скоростью $p[j] < p[i]$. Другими словами, выбранные для ускорения планеты должны образовывать **возрастающую подпоследовательность** чисел $p[0], p[1], \ldots, p[n-1]$. Подпоследовательность последовательности p представляет собой последовательность, полученную из p удалением нуля или более элементов p. Например, $[0]$, $[]$, $[0, 2]$ или $[0, 1, 2]$ являются подпоследовательностями $[0, 1, 2]$, а $[2, 1]$ не является.

Ученые посчитали, что у корабля есть ровно k различных способов выбрать искомое множество планет, используемых для ускорения, однако, они потеряли доступ к орбитальным скоростям планет (и даже к значению n). Однако они помнят, $(p[0], p[1], \ldots, p[n-1])$ представляет собой перестановку чисел $0, 1, \ldots, n-1$. Перестановкой называется последовательность целых чисел от 0 до $n-1$, в которой каждое число встречается ровно 1 раз. Ваша задача состоит в том, чтобы восстановить последовательность $p[0], p[1], \ldots, p[n-1]$ с достаточно небольшим n.

Вам необходимо решить задачу для q различных кораблей. Для каждого корабля i вам будет дано число k_i, обозначающее число способов выбрать планеты для ускорения корабля. Ваша задача состоит в том, чтобы найти последовательность с достаточно небольшим значением n_i, для которой существует ровно k_i способов выбрать подпоследовательность планет с возрастающими орбитальными скоростями.

Implementation details

Вам необходимо реализовать следующую функцию:

```
int[] construct_permutation(int64 k)
```

- k: требуемое число возврастающих подпоследовательностей.
- Данная функция должна возвращать массив из n чисел, каждое из которых должно находиться в диапазоне между 0 и $n-1$ включительно.

- Возвращаемый массив должен образовывать корректную перестановку с ровно k возрастающими подпоследовательностями.
- Данная функция будет вызвана q раз. Каждый из таких вызовов должен обрабатываться независимо.

Constraints

- $1 \leq q \leq 100$
- $2 \leq k_i \leq 10^{18}$ (для всех $0 \leq i \leq q - 1$)

Subtasks

1. (10 баллов) $2 \leq k_i \leq 90$ (для всех $0 \leq i \leq q - 1$)
2. (90 баллов) Без дополнительных ограничений. Пусть m обозначает максимальную длину перестановки, полученной вашим решением, для всех возможных кораблей в этой подзадаче. Тогда ваш балл вычисляется согласно следующей таблице:

Длина перестановки	Баллы
$m \leq 90$	90
$90 < m \leq 120$	$90 - \frac{(m-90)}{3}$
$120 < m \leq 5000$	$80 - \frac{(m-120)}{65}$
$m > 5000$	0

Example

Example 1

Рассмотрим следующий вызов функции:

```
construct_permutation(3)
```

Функция должна вернуть перестановку с ровно 3 возрастающими подпоследовательностями. Один из возможных ответов — $[1, 0]$, для которой $[]$ (пустая подпоследовательность), $[0]$ и $[1]$ являются ее возрастающими подпоследовательностями.

Example 2

Рассмотрим следующий вызов функции:

```
construct_permutation(8)
```

Функция должна вернуть перестановку с ровно 8 возрастающими подпоследовательностями. Один из возможных ответов — $[0, 1, 2]$.

Sample grader

Грейдер считывает данные в следующем формате:

- строка 1: q
- строка $2 + i$ ($0 \leq i \leq q - 1$): k_i

Грейдер выводит в отдельной строке результат вызова `construct_permutation` или сообщение об ошибке.