Задача A1. Единици

⏳1.0 sec. 💾256 MB

Радко отново иска да разбере редицата $p_1, p_2, ..., p_n$ на Марти. Този път Марти реши да бъде по-полезен и директно да каже, че редицата се състои от n бита 0 и 1, като точно k от тях са 1. Този път той ще отговаря само на следния въпрос:

- "Има ли единица измежду $p_l, p_{l+1}, ..., p_r$?"

За съжаление, Радко все още е твърде зает и отново аутсорсва задачата на Вас. Програмата ви ще бъде тествана с n_{tests} подтеста за всеки тест и резултатът Ви ще бъде изчислен спрямо общия брой на въпросите, с които откривате поредните редици.

Детайли по имплементацията

Вашата функция `guessOnes` има следния прототип:

```
std::vector<int> guessOnes(int n, int k);
```

Тя ще бъде извикана n_{tests} пъти за всеки тест и ще получи като аргумент размера на редицата n и броя единици k. Функцията трябва да върне вектор от k числа – позициите на единиците в нарастващ ред.

Функция $hasOnes$ на журито има следния прототип:

```
bool hasOnes(int l, int r);
```

Вашата програма може да я вика колкото пъти иска. Като аргументи се подават два индекса l и r от 1 до n, за които искате да зададете въпрос. Функцията връща дали има единица измежду $p_l, p_{l+1}, ..., p_r$. Тя работи със сложност $O(1)$.

Вашата програма трябва да имплементира функцията `guessOnes`, но не трябва да съдържа функция `main`. Освен това, тя трябва да не чете от стандартния вход или да печата на стандартния изход. Програмата ви също така трябва да включва хедър файла `ones.h` чрез указание към предпроцесора: `#include "ones.h"`

Стига да спазва тези условия, програмата ви може да съдържа каквито и да е помощни функции, променливи, константи и прочее.

Ограничения

- Всяка редица е произволно генерирана.
- $n = 100000$
- $n_{tests} = 100$

Подзадачи и оценяване

Частта от точките, които ще получите на дадена подзадача зависи от общия брой въпроси, които задавате на подтест, $q_{participant}$, и от константата за подзадачата q_{author}.

Ако $q_{participant} \leq q_{author}$, $score = 1$

Иначе $score = 1 - \sqrt[4]{1 - \frac{q_{author}}{q_{participant}}}$

Подзадачи

Подзадача	**Точки**	k	q_{author}
1	10	10	14480
2	10	20	27201
3	10	50	61908
4	10	100	114144
5	10	200	208697
6	10	500	455937
7	10	1000	811792
8	10	2000	1422396
9	10	5000	2879675
10	10	10000	4723779

Локално тестване

В системата ви е предоставен файлът `Lgrader.cpp`, чрез който може да тествате локално програмата си. За целта трябва да добавите `#include "Lgrader.cpp"` към кода си.

На първия ред на стандартния вход се въвеждат числата n, k и n_{tests}.

Следват n_{tests} теста, за всеки от които се въвеждат по k числа – позициите на единиците. Ако програмата Ви успешно намери правилната редица за всеки тест, накрая ще се изведе общия брой заявки, които сте използвали за всички редици.