Задача 3. Sorting

Сашка редовно участва в онлайн състезания по програмиране. Наградата в текущото е участие на лагер-школа по информатика на Малдивите. Специално за него тя си е избрала кодовото име **Sorting**. Една от задачите е следната:

Журито е намислило пермутация $p_1, p_2, p_3, ..., p_N$ на числата от 1 до N. Задачата е да се познае пермутацията, като за целта към журито могат да се задават следните два вида въпроси:

1) За дадени две позиции x и y в пермутацията ($1 \leq x, y \leq N$), дали $p_x < p_y$.
2) За дадено число d и две позиции x и y в пермутацията ($1 \leq x, y, d \leq N$), дали $|p_x - p_y| \equiv 0 (mod\ d)$.

Иначе казано, дали разликата на елементите в пермутацията на позиция x и позиция y се дели на d.

От първия вид трябва да се задават колкото е възможно по-малко въпроси (вижте оценяването по-долу), докато броят въпроси от втория вид е неограничен.

Задача

Помогнете на Сашка, като напишете програма `sorting`, която по дадено N възстановява пермутацията. Тя трябва да съдържа функцията *`solve`*, която ще се компилира и изпълнява с програма на журито.

Детайли по имплементацията

Функцията *`solve`* трябва да има следния формат:

```
std::vector <int> solve(int N);
```

Функцията се извиква с един параметър *`N`*, равен на броя елементи в пермутацията. Тя трябва да върне вектор с точно N различни елемента, съответно пермутацията, която програмата Ви е намерила.

За въпросите към журито са предоставени функциите *`compare`* и *`divisible`*.

Функцията *`compare`* има следния формат:

```
bool compare(int x, int y);
```

Функцията връща `true`, ако $p_x < p_y$ и `false` в противен случай.

Функцията *`divisible`* има следния формат:

```
bool divisible(int x, int y, int d);
```

Функцията връща `true`, ако $|p_x - p_y| \equiv 0\ (mod\ d)$ и `false` в противен случай.

Ако дадете невалидни параметри на функциите, програмата Ви ще се прекъсне и ще получите `Wrong Answer`.

Вие трябва да предадете към системата файл `sorting.cpp`, който съдържа функцията *`solve`*. Той може да съдържа и друг код, и функции, необходими за работата Ви, но не трябва да съдържа главната функция *`main`*. Също така, не трябва да четете от стандартния вход или да пишете на стандартния изход. Програмата Ви също така трябва да включва хедър файла `sorting.h` чрез указание към препроцесора:

```
#include "sorting.h"
```

Ограничения

$2 \leq N \leq 500\,000$

$1 \leq p_i \leq N$ за всяко $1 \leq i \leq N$.

$p_i \neq p_j$ за всички $1 \leq i < j \leq N$.

Оценяване

В тестове, носещи 20% от точките: $N \leq 500$
В тестове, носещи 30% от точките: $N \leq 2000$
В тестове, носещи 40% от точките: $N \leq 10\,000$
В тестове, носещи 60% от точките: $N \leq 75\,000$
В тестове, носещи 100% от точките: $N \leq 500\,000$

Ако не сте познали пермутацията, Вие ще получите 0% от точките на съответния тест. В противен случай, нека сте питали Q въпроса от първия вид. Тогава се определя стойност на променливата P по следния начин:

- Ако $0 \leq Q \leq \frac{N}{2}$, $P = 1.00$.
- Ако $\frac{N}{2} < Q \leq N$, $P = 0.70$.
- Ако $N < Q \leq 2N$, $P = 0.50$.
- Ако $2N < Q \leq N\lceil log_2 N \rceil$, $P = 0.15$.
- Ако $N\lceil log_2 N \rceil < Q \leq N(1 + \lceil log_2 N \rceil)$, $P = 0.10$.
- Ако $N\left(1 + \lceil log_2 N \rceil\right) < Q$, $P = 0.05$.

Тогава, точките, които ще получите за теста, са равни на: $(\text{точките\_за\_теста}).P$

Примерна комуникация

Функция на състезателя	Програма на журито
	Извиква **solve(3)**
Извиква **compare(1,2)**	Връща стойност **true**
Извиква **compare(1,3)**	Връща стойност **false**
Извиква **divisible(1,3,2)**	Връща стойност **false**
Връща стойност **{2,3,1}**	

Локално тестване

За локално тестване са предоставени файловете `sorting.h` и `Lgrader.cpp`. Сложете Вашия файл `sorting.cpp` и двата предоставени файла в една папка. Като компилирате заедно трите файла ще получите програма, с която ще проверите верността на функцията Ви. Програмата ще изисква от стандартния вход следната последователност от данни:

- на първия ред: едно цяло положителни число – броят числа в пермутацията N.

- на втория ред: N различни цели положителни числа, всяко със стойност между 1 и N – намислената от журито пермутация.

Ако направите невалидно извикване или намерите грешно пермутацията, то ще получите подходящо съобщение на изхода. Иначе на изхода ще получите съобщение “`Correct!`” и стойността на променливата P, изчислена, както е описано по-горе.