**Задача А2. Shoes**

Матю има на брой обувки, номерирани с числата от до . Те образуват чифта. Обуквите обаче не са подредени по чифтове, а са разбъркани произволно. Вие искате да намерите всички чифтове, но Матю ви помага само като отговаря на следния въпрос:

* Има ли в множеството обувки поне един чифт?

Матю се измаря бързо, затова целта ви е да намалите броя на въпросите, които го питате.

Програмата ви ще бъде тествана на подтеста за всеки тест и резултатът Ви ще бъде изчислен спрямо средния брой заявки за подтест.

**Детайли по имплементацията**

Вашата функция guessPairs има следния прототип:

std::vector<std::pair<int, int>> guessPairs(int n);

Тя ще бъде извикана пъти за всеки тест и ще получи като аргумент броя чифтове обувки. Функцията трябва да върне вектор от двойки числа – номерата на обувките във всеки чифт. Подредбата на двойките във вектора или числата във двойките нямат значение.

Функция pairInSet на журито има следния прототип:

bool pairInSet(const std::vector<int> &v);

Вашата програма може да я вика колкото пъти иска. Като аргумент ѝ се подава константна референция към вектор, който съдържа номерата на обувките, за които искате да зададете въпрос. Тези номера трябва да са различни числа от до . Функцията връща *true* ако има поне един чифт в множеството, и *false* иначе. Тя работи със сложност , където е дължината на вектора.

Вашата програма трябва да имплементира функцията guessPairs, но не трябва да съдържа функция main. Освен това, тя трябва да не чете на стандартния вход или да печата на стандартния изход. Програмата ви също така трябва да включва хедър файла **shoes.h** чрез указание към предпроцесора:

#include “shoes.h”

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

**Ограничения**

Обувките са разбъркани произволно. (В първата подзадача левите и десните обувки са разбъркани поотделно)

**Подзадачи и оценяване**

Частта от точките, които ще получите на дадена подзадача зависи от средният брой заявки, които правите на подтест, , и от константите за подзадачата и .

Ако :

Иначе:

|  |  |  |  |
| --- | --- | --- | --- |
| Подзадача | Точки | Допълнителни ограничения | Константи за оценяване |
| 1 | 10 | Обувки с номера са леви, а  са десни.  Всеки чифт се състои от една лява и една дясна обувка. |  |
| 2 | 90 | Няма |  |

**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 и .

Следват теста, за всеки от които се въвеждат по двойки числа – номерата на обувките във всеки чифт.

Ако програмата Ви успешно намери правилните чифтове за всеки тест, накрая ще се изведе средния брой на заявки, които сте използвали.