

Task 1. Heavy coins

В низменном королевстве была величественная стена, веками защищавшая его территорию от вторжений. Однако ровно год назад банда грабителей решила, что стена усложняет их деятельность, и сожгла ее дотла. Теперь принцесса Мария планирует нанять опытных строителей, чтобы вернуть стене ее первоначальный вид. Чтобы платить строителям, она пошла в королевскую сокровищницу и нашла $N = 2^K$ монет, каждая из которых имеет уникальный номер от 0 до $N - 1$. Королевский казначей проинформировал ее, что существует два типа монет – легкие и тяжелые, при этом все тяжелые монеты имеют одинаковый вес, аналогично все легкие монеты тоже. Также известно, что в сокровищнице существует по крайней мере одна тяжелая монета и по крайней мере одна легкая.

По условиям, согласованным со строителями новой стены, княгиня Мария должна платить им только в тяжелых монетах, а до начала работ она должна аванс в одну тяжелую монету. Проблема в том, что казначей не знает, сколько в сокровищнице тяжелых монет, и какие именно монеты тяжелые. К счастью, у него есть весы, с помощью которых он может сравнить общий вес двух наборов, состоящих из одинакового количества монет. Теперь казначей должен удостовериться, что количества тяжелых монет в королевской сокровищнице достаточно для выплаты, и должен найти одну такую монету, которой можно оплатить аванс. Однако это отняло бы у него слишком много времени, а принцесса Мария хочет, чтобы строительство новой стены началось как можно скорее, поэтому она обращается к вам за помощью.

Задача

Напишите программу `coins`, реализующую функцию `count_heavy`, которая будет взаимодействовать с программой жюри и в результате этого взаимодействия определит общее число тяжелых монет в королевской казне, а также номер одной такой монеты.

Детали реализации

Ваша функция должна иметь следующий формат:

```
pair<int, int> count_heavy(int k)
```

Она получает в качестве параметра число K (напомним, что общее число монет равно 2^K) и должна вернуть пару целых чисел – количество тяжелых монет и номер одной из таких монет.

Для взаимодействия с программой жюри вам предоставляется следующая функция:

```
int weigh(const vector<int> &a, const vector<int> &b)
```

Она получает в качестве параметров два вектора, содержащих номера монет для взвешивания. Векторы должны иметь одинаковое количество элементов и не должны содержать повторяющихся номеров. Функция будет возвращать одно из следующих значений:

- ☐ 0, если два набора монет имеют одинаковый общий вес;
- ☐ -1, если общий вес монет, чьи номера записаны в векторе a , меньше, чем общий вес монет, чьи номера записаны в b ;
- ☐ +1, если общий вес монет, чьи номера записаны в векторе a , больше, чем общий вес монет, чьи номера записаны в b .

Заметим, что сложность выполнения функции `weigh` пропорциональна общему числу элементов в переданных ей векторах.

Вы должны посылать на проверку в систему файл `coins.cpp`, который содержит функцию `count_heavy`. Он может содержать другой код и функции, необходимые вам для работы, но он не должен содержать функцию `main`. Также вы не должны читать из стандартного ввода или выводить в стандартный поток вывода. Ваша программа также должна включать `coins.h` заголовочный файл в инструкцию для препроцессора:

```
#include "coins.h"
```

Ограничения

$$1 \leq K \leq 20$$

Scoring

Если в процессе выполнения ваше решение выдает корректный ответ на тест, используя не более чем 10 000 вызовов функции *weigh* function, коэффициент для баллов за тест вычисляется по формуле:

$coef = \min\left(1, \frac{11}{9} * \left(\frac{author+1}{yours+1}\right)\right)$, где *author* – количество вызовов функции *weigh* function, которое осуществляет на этом тесте одно из авторских решений, а *yours* – количество вызовов функции *weigh*, выполняемых вашим решением. В противном случае за этот тест *coef* будет иметь значение 0.

Гарантируется, что во всех тестах значение *author* строго меньше, чем 1 000.

Каждая подзадача содержит определенное число групп, каждая группа содержит ровно три теста. Каждая группа оценивается **независимо**, и количество баллов, которые вы получите за группу, равно произведению количества баллов за группу на минимальное значение коэффициента *coef* среди тестов этой группы.

Subtasks

Subtask	Points	Other constraints
1	5	$K \leq 5$
2	10	В каждом тесте или ровно одна тяжелая или ровно одна легкая монета.
3	20	В каждом тесте не более K тяжелых или не более K легких монет.
4	25	$K \leq 10$
5	40	Нет дополнительных ограничений.

Пример взаимодействия

Функция участника	Программа жюри
	Вызывает <code>count_heavy(2)</code>
Вызывает <code>weigh({0, 1}, {2, 3})</code>	Возвращает значение -1
Вызывает <code>weigh({1, 2}, {0, 3})</code>	Возвращает значение +1
Вызывает <code>weigh({1}, {2})</code>	Возвращает значение 0
Возвращает {3, 2}	

Локальное тестирование

Для локального тестирования вам предоставляется файл `coins.h` и `Lgrader.cpp`. Разместите ваш файл `coins.cpp` и два предоставляемых файла в одной папке. Затем скомпилируйте три файла вместе. Таким образом вы получите программу, позволяющую проверить корректность вашей функции. Программу будет требовать следующую последовательность данных:

- в первой строке одно положительное целое число, обозначающее значение K .
- во второй строке 2^K чисел со значениями 1 или 0, обозначающими тяжелые и легкие монеты соответственно.

Программа будет выводить информацию о вызовах функции *weigh*, ответ, который ваша функция `count_heavy` возвращает, и сообщение – корректный это ответ или нет.