

Task 1. Heavy coins

У нижньому королівстві колись була висока стіна, яка віками захищала територію від загарбників. Але, рік тому, група бандитів-рецидивістів вирішила що стіна занадто заважає їм займатись контрабандною діяльністю, і тому вони повністю спалили її, до самісінької землі. І ось тепер, принцеса Марічка зажадала найняти найдосвідченіших будівельників у всьому королівстві, аби вони допомогли відбудувати стіну і повернути їй колишню славу і міць. Щоб платити будівельникам, Марічка звернулася до королівської скарбниці й отримала $N = 2^K$ монет, кожна з яких мала унікальний номер від 0 до $N - 1$. Дідусь-Скарбник попередив Марічку що монети бувають двох типів — важкі та легкі, всі важкі монети мають однакову масу, так само легкі монети — між собою всі рівної маси. Також він запевнив що серед виданих монет є хоча б одна легка і хоча б одна важка.

Згідно з угодою, закладеною з будівельниками, Марічка буде платити їм тільки важкими монетами. Перед початком роботи робітники потребують аванс — одна важка монета. Але дідусь Скарбник вже досить старенький, і не пам'ятає які з монет є якими, ба більше, він навіть забув скільки монет яких типів там є, і монет може взагалі не вистачить! На щастя у нього залишились старі терези, які можуть порівняти два набори монет, але з умовою що кількість монет в обох наборах однакова. Доведеться за їх допомогою знаходити важкі монети... Але оскільки розпочати побудову муру треба негайно, то Марина просить Вас віднайти кількість важких монет і знайти одну важку монету, щоб сплатити аванс якомога швидше.

Завдання

Напишіть програму *coins*, в якій реалізуйте функцію *count_heavy*, котра буде скомпільована з програмою журі й, через комунікацію з якою, знаходить загальну кількість важких монет і номер однієї з них.

Деталі реалізації

Ви маєте реалізувати функцію, яка має наступний формат наступний формат:

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

Вона отримує як параметр номер K (нагадуємо, кількість монет 2^K) і повинна повернути пару двох чисел — число важких монет і номер однієї з них.

Для комунікації з програмою журі Ви можете викликати наступну функцію:

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

Вона приймає як параметр два вектори (структура *std::vector* в c++) — номери монет які будуть покладені відповідно на першу і другу шальку терезів. Ці вектори мають мати однаковий розмір, і всі номери з обох векторів мають бути попарно різні між собою (адже не можна одну монету покласти на терези двічі). Ця функція може повернути наступні значення:

- 0, якщо набори важать однаково, тобто сумарна маса монет на першій шальці рівна сумарній масі на другій шальці.
- - 1, якщо сумарна маса монет з номерами з вектора *a* менша ніж маса монет з номерами з *b*;
- + 1, якщо сумарна маса монет з номерами з вектора *a* менша ніж маса монет з номерами з *b*;

Зауважте, що часова складність функції *weigh* пропорційна кількості елементів у векторах які є параметрами функції.

Ви маєте надіслати до тестувальної системи файл *coins.cpp*, яка містить функцію *count_heavy*. Також файл може містити інший код, змінні, функції, які необхідні Вам для роботи, але не має містити функції *main*. Також Вам не дозволяється нічого читати зі стандартного потоку або виводити туди щось. Також Ви повинні включити файл заголовка *coins.h*, це можна зробити наступною командою:

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

Обмеження

$$1 \leq K \leq 20$$

Якщо протягом виконання ваша програма зробить не більше ніж 10 000 запитів до функції *weight*, то деякий коефіцієнт буде розрахований за формулою $coef = \min\left(1, \frac{11}{9} * \left(\frac{author+1}{yours+1}\right)\right)$, де *author* — це максимальна кількість викликів функції *weight* в одного з рішення авторів (див. підзадачі нижче), а *yours* — кількість викликів функції *weight* у Вашого рішення. Інакше *coef* отримає значення 0.

Кожна підзадача складається з певної кількості груп, кожна група містить рівно три тести. Кожна з груп буде оцінюватися **незалежно** і кількість балів, отриманих за неї, буде рівна добутку кількості балів які їй виділено і мінімальному значенню *coef* по всіх тестах в цій групі.

Підзадачі

Підзадача	Бали	<i>author</i>	Додаткові обмеження
1	5	≤ 40	$K \leq 5$
2	10	≤ 250	Існує тільки одна легка або тільки одна важка монета.
3	20	≤ 1000	Існує не більше ніж <i>K</i> важких монет або не більше ніж <i>K</i> легких.
4	25	≤ 100	$K \leq 10$
5	40	≤ 500	Без додаткових обмежень.

Приклад комунікації

Функція учасника	Програма журі
	Викликає <i>count_heavy(2)</i>
Викликає <i>weigh({0, 1}, {2, 3})</i>	Повертає значення -1
Викликає <i>weigh({1, 2}, {0, 3})</i>	Повертає значення +1
Викликає <i>weigh({1}, {2})</i>	Повертає значення 0
Повертає {3, 2}	

Тестування на Вашій системі

Для того, щоб перевірити свою програму локально, Вам надано файли *coins.h* і *Lgrader.cpp*. Помістіть Ваше розв'язання *coins.cpp* і два наданих файли в одну директорію. Потім скомпілюйте їх разом. На виході отримаєте програму, щоб перевірити правильність Вашого рішення. Програма зчитує таку послідовність даних:

- У першому рядку зчитується значення *K*.
- У другому рядку зчитується 2^K чисел зі значеннями 1 або 0, яке означає що монета важка або легка, відповідно.

Ця програма виведе інформацію про виклики до функції *weigh*, відповідь, яку повернула Ваша функція *count_heavy* і висновок, який інформує, чи правильна Ваша відповідь, чи ні.