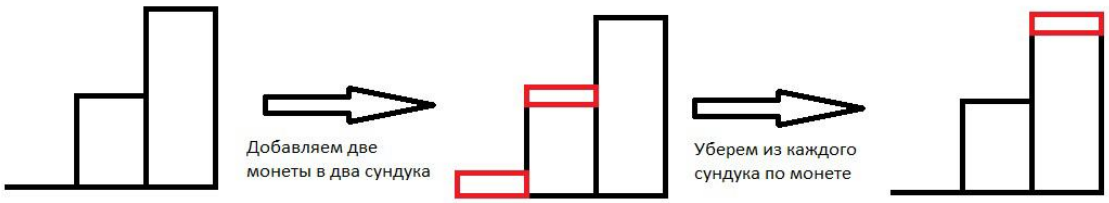
**Задача A. Прибытие короля**

Отсортируем сундуки по возрастанию так, чтобы в первом было минимальное число монет, а в последнем максимальное. Уменьшим количество монет во всех сундуках на количество монет в первом сундуке. Заметим, что кинуть по монете в два сундука – это тоже самое, что и забрать одну монету из третьего сундука:




Значит если мы хотим уравнять количество монет в трех сундуках, мы должны убрать лишние монеты (сделать количество монет в сундуках равным нулю). Ответом будет разность между суммой монет в изначальных сундуках и утроенным минимумом количества монет, содержащихся в этих сундуках.

Пример реализации этого решения на языке C++:

```
#include <iostream>

using namespace std;

long long a, b, c;

int main() {
    cin >> a >> b >> c;
    cout << (a + b + c) - 3 * min(a, min(b, c));
    return 0;
}
```

**Задача B. Кассовый разрыв**

Идея решения: расположим выплаты, уменьшающие счет, как можно раньше, а увеличивающие счет – как можно позже. Докажем, что если в какой-то последовательности выплат мог случиться кассовый разрыв, то в такой последовательности он тоже обязательно случится.

**Доказательство**: рассмотрим последовательность выплат, в которой случается кассовый разрыв. Рассмотрим самую раннюю операцию, стоящую не на том месте, на котором мы хотим ее расположить.

1. Если это операция, уменьшающая деньги на счёте компании, то после ее перемещения в начало диапазона, в каждый момент времени на счёте будет не больше денег, чем в исходной последовательности операций, и кассовый разрыв случится не позже
2. Если это операция, увеличивающая деньги на счёте компании, то аналогично при перемещении ее в конец доступного интервала, в каждый момент времени на счёте будет не больше денег, чем в исходной последовательности, и кассовый разрыв произойдет в тот же момент или раньше.

Таким образом, чтобы проверить, что кассовый разрыв может произойти, достаточно поставить все операции, уменьшающие счёт, в начало их возможного диапазона, а увеличивающие – в конец.

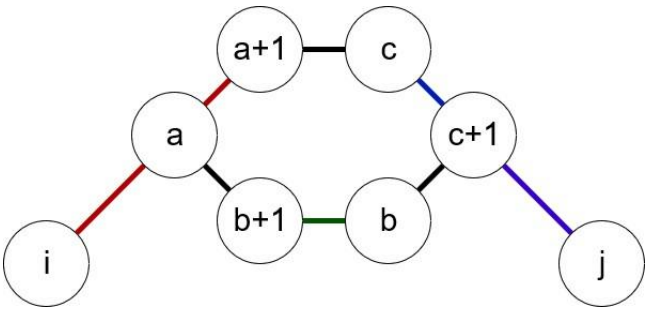
Если теперь упорядочить все операции по времени, при этом операции внутри одного дня упорядочить так, чтобы уменьшающие шли раньше увеличивающих, то кассовый разрыв равносилен тому, что все операции до какого-то момента времени суммарно уменьшают счёт больше, чем на $s$. Для этого достаточно просто пройтись по операциям, упорядоченным по времени, и в каждый момент посчитать текущий счёт.

Ограничения задачи позволяли воспользоваться любой сортировкой для упорядочивания событий, и также считать суммарный счёт в каждый момент времени за $O(m)$ или $O(n)$.

### Задача C. Метро

У данной задачи есть множество решений, рассмотрим два различных подхода:

1. Можно заметить, что в данной задаче есть ровно восемь «интересных» станций (стартовая и конечная станции, а также двойные станции пересадок).




Можно построить граф на в котором вершины соответствуют станциям и есть следующие ребра:

- Ребра веса $d$, между станциями пересадок
- Ребра веса $t_i$, между соседними станциями одной линии
- Ребра между стартовой вершиной и вершинами пересадок на ее линии (вес считается, как разность номеров вершин умножить на соответствующее $t_i$)
- Аналогично ребра между конечной вершиной и вершинами пересадок на ее линии
- Ребро между стартовой и конечной вершинами, если они располагаются на одной линии

Затем на построенном графе можно запустить любой из алгоритмов поиска кратчайших путей, например алгоритм Флойда, чтобы найти кратчайшее расстояние между вершинами, соответствующими начальной и конечной станциям.

2. Можно же было по аналогичному рисунку понять, что существует всего четыре способа движения от стартовой станции до конечной:

• Стартовая и конечная станция на одной линии:

– Ноль пересадок

– Три пересадки

• Стартовая и конечная станция на разных линиях:

– Одна пересадка

– Две пересадки

После осознания набора этих случаев, остается лишь аккуратно рассмотреть данные случаи и выбрать минимальный.

P.S. Нужно не забыть про 64-битный тип данных независимо от способа решения.

**Задача D. Проверка теста**

Нужно для каждой пары работ проверить, являются ли они похожими. Чтобы это сделать, сначала посчитаем количество правильных ответов в каждой работе. Потом для каждой пары работ посчитаем количество одинаковых правильных ответов и количество одинаковых неправильных ответов. Теперь можно проверить условие того, что эти работы похожи.

Посчитаем асимптотику этого решения: решение рассматривает $\frac{m \cdot (m-1)}{2} = O(m^2)$ пар работ, для каждой пары решение просматривает ответы в этих работах на все $n$ вопросов, поэтому асимптотика суммарного времени работы равна $O(m^2 \cdot n)$.

**Задача E. Магический фокус**

Выберем произвольную тройку и переберём все варианты её упорядочить.

Будем поэлементно восстанавливать последовательность. Рассмотрим все тройки, содержащие два последних элемента последовательности. Так как входные данные корректны, то их не более двух – тройка из последних трёх элементов или тройка из последних двух и следующего элемента.

Если можем, выписываем следующий элемент. Иначе рассматриваем другой способ упорядочить изначальную тройку.

Как по паре элементов ($a$, $b$) сохранить подходящие третьи элементы из троек? Воспользуемся `std::map<pair<int, int>, vector<int>>`, или её аналогами.

Следует разобрать отдельно крайними случаи $n \leq 4$, в таком случае ответом является произвольная перестановка.