Задание 1
Точки в линию

ограничение по времени на тест
3 секунды
ограничение по памяти на тест
15 мегабайт
ввод
стандартный ввод
вывод

Кубик указал на плоскости точки с целочисленными координатами. Он захотел узнать какое максимальное количество точек лежит на одной прямой. Помогите Кубику найти это количество

Входные данные:
Первая строка содержит целое число N ($0<N<10^3$) –количество точек.
Вторая строка содержит N пар целых чисел (x_1, y_1), (x_2, y_2), …, (x_N, y_N) ($1 \le x_i, y_i \le 10^5$), где (x_i, y_i)— координата i-й точки.

Выходные данные:
Выведите максимальное количество точек, которые лежит на одной прямой.

Пример
входные данные
```
7
0 0
0 6
6 6
6 0
3 3
3 8
-1 -1
```
выходные данные
```
4
```

Решение на языке C++:
```
#include <iostream>
#include <cmath>
using namespace std;

int x[1000], y[1000];
int n;

int main() {
	cin >>n ;
	for (int i = 0; i < n; ++i) {
		cin >> x[i] >> y[i];
	}
```

```
	int result = 0;
	for (int i = 0; i < n; ++i)
		for (int j = i + 1; j < n; ++j) {
			int r = 2;
			for (int k = j + 1; k < n; ++k) {
				if (x[i] == x[j]) {
					if (x[i] == x[k])
						r++;

				}
				else if (y[i] == y[j]) {
						if (y[i] == y[k])
							r++;

					}
					else {
					if ((y[k]-y[i])*(x[j]-x[i]) == (y[j] - y[i]) * (x[k] - x[i]))
						r++;
					}
			}
			if (r > result)
				result = r;

		}
	cout << result;
	return 0;
}
```

Задание 2
Тяжелые грибы

ограничение по времени на тест
30 секунд
ограничение по памяти на тест
16 мегабайт
ввод
стандартный ввод
вывод
стандартный вывод

Кубик пошел по грибы. Он не очень сильный, а грибы очень тяжелые. Кубик может нести только по одному грибу в каждой руке. В правой руке он может нести гриб веса не более K, а в левой – не более T. В процессе прогулки по лесу он нашел N грибов весами $a_1, a_2, \ldots, a_N$.
Определите максимальный суммарный вес грибов, которые Кубик сможет принести домой, или 0 если он ничего не принес.

Входные данные:
Первая строка содержит три целых числа K, T, N ($0<K, T, N<10^7$) – максимальный вес который может нести Кубик в левой и правой руке и количество найденных грибов.
Вторая строка содержит N целых чисел $a_1, a_2, \ldots, a_N$ ($1 \le a_i \le 10^{15}$), где a_i — это вес i-го гриба.

Выходные данные:
Выведите максимальный суммарный вес грибов, которые Кубик сможет принести домой, или 0 если такого гриба нет.

Пример
входные данные
10 8 5
18 3 22 9 14
выходные данные
12

Решение на языке C++:

```
#include <iostream>
using namespace std;

int main() {
	long long t, k, n, a, rt,rk;
	rt = 0;
	rk = 0;
	cin >>k>>t>>n ;
	if (k < t) {
		a = k;
		k = t;
		t = a;
```

```
	}
	rt = 0;
	rk = 0;
	for (long long i = 0; i < n; ++i) {
		cin >> a;
		if (a <= k && rk < a) {
			if (rk <= t && rk > rt) rt = rk;
			rk = a;
		}
		else
			if (a <= t && rt < a) rt = a;
	}
	cout << rk + rt;
	return 0;
}
```

Задание 3
Красивый букет

ограничение по времени на тест
20 секунд
ограничение по памяти на тест
256 мегабайт
ввод
стандартный ввод
вывод

Кубик решил собрать букет цветов. Кубик считает букет красивым, количество цветов в букете кратно *M*. Кубик собирал цветы на полянках. Известно, что Кубик либо собирал все цветы только одного вида с полянки, либо не брал их вовсе. Всего в лесу *N* полянок и *K* видов цветов.
Определите максимальное количество цветов в красивом букете, который Кубик может собрать.

Входные данные:
Первая строка содержит три целых числа *M, N, K* ($0 \le N \le 10^5$, $0 \le K, M \le 100$) – кратность числа цветов в букете, количество полянок и количество различных цветов в лесу
Затем следует *N* строк, в каждой строке записано *K* целых чисел: $a_1, \ldots, a_K$ ($1 \le a_i \le 10^4$)— количество цветов каждого вида на очередной полянке.
.

Выходные данные
Выведите целое число – максимальное количество цветов в букете.

Примеры
входные данные
10 5 3
1 5 8
9 4 7
4 6 2
4 3 1
8 1 6
выходные данные
30

Решение на языке C++:

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
	int m[100];
```

```
	for (int i = 0; i < 100; ++i) m[i] = 0;
	int M,N,K;
	cin >> M >> N >> K ;
	for (int g= 0; g < N; ++g) {
		int mm[100];
		for (int i = 0; i < M; ++i) mm[i] = m[i];
		for (int k = 0; k < K; ++k) {
			int a;
			cin >> a;
			for (int i = 0; i < M; ++i)
				if (m[i]>0)
					mm[(i + a) % M] = max(mm[(i + a) % M], m[i] + a);
			mm[a % M] = max(a, mm[a % M]);
		}
		for (int i = 0; i < M; ++i) {
			m[i] = mm[i];
		}
	}
	cout << m[0];
	return 0;
}
```

Задание 4

Восстановить мосты

ограничение по времени на тест: *10 секунд*
ограничение по памяти на тест: *64 мегабайта*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Страна Кубикляндия представляет собой группу островов, часть из которых соединены мостами. Для установления сообщения между всеми островами необходимо построить дополнительные мосты. Для каждого нового моста известна его цена. Президент Кубикляндии решил потратить минимальное количество денег для построения мостов, при этом необходимо чтобы из любого острова можно попасть на любой другой остров.

Помогите президенту установить связь между всеми островами страны, при этом потратив минимальное количество денег.

Входные данные:

Первая строка содержит три целых числа *n, m, d* ($0<n\leq 2*10^3$; $1\leq m, d \leq 3*10^4$, $m>d$) – количество островов, количество существующих мостов и количество мостов, которые можно построить.

Все острова перенумерованы числами от *1* до *n*.

В следующих *m* строках записаны пара чисел *v1, v2* ($0<v1, v2 \leq n$)– номера островов, соединенных мостом.

В следующих *d* строках записаны тройка чисел *v1, v2, cost* ($0<v1, v2 \leq n$) ($0< cost \leq 10^5$) – номера островов, мост между которыми можно построить, и стоимость постройки этого моста.

Выходные данные

Выведите целое число — минимальную стоимость установления связи между всеми островами страны.

Примеры

входные данные

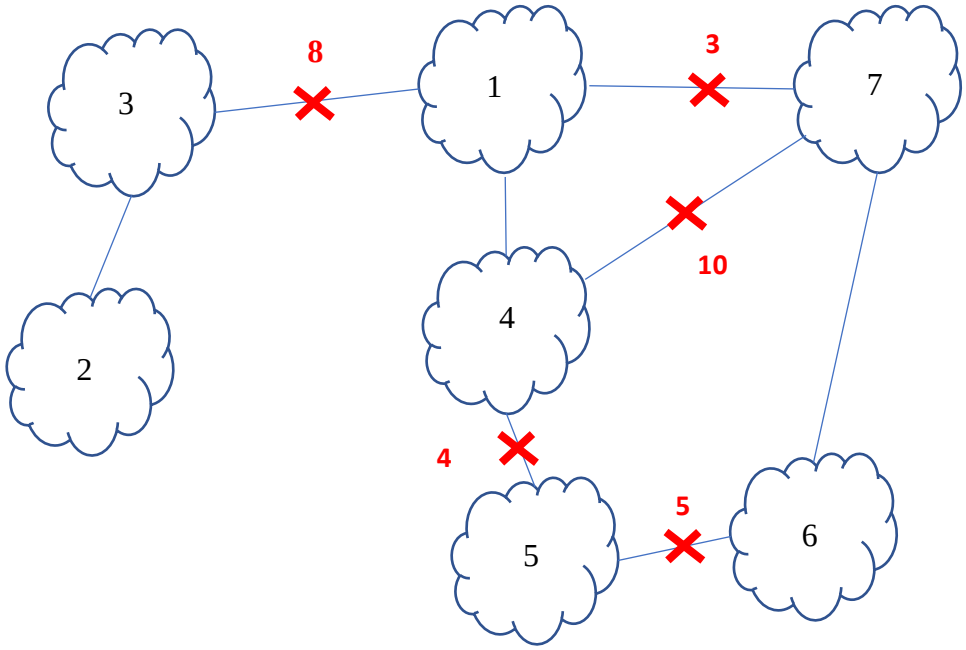
```
7 3 5
2 3
1 4
7 6
1 3 8
1 7 3
4 7 10
4 5 4
5 6 5
```

выходные данные

```
15
```

Примечание

На рисунке мосты, которые можно построить, обозначены крестиком, а значение около крестика – стоимость постройки. При построении мостов (1, 3), (1, 7) и (4, 5) суммарная стоимость будет минимальна.



Решение на языке C++:

```
#include <iostream>
#include <cmath>
using namespace std;

struct bridge {
    int v1, v2, cost;
};

int connected[2001];
bridge forBuild[30001];
int n, m, d, i, j, v1, v2, c;
long long answer;

void sortBridge() {
    for (int k = d; k > 0; k--)
        for (int i = 1; i < k; ++i)
            if (forBuild[i - 1].cost > forBuild[i].cost) {
                bridge b = forBuild[i - 1];
                forBuild[i - 1] = forBuild[i];
                forBuild[i] = b;
            }
}

int getComponent(int a) {
    if (connected[a] == a) return a;
```

```
    return connected[a] = getComponent(connected[a]);
};
int main() {
    cin >> n >> m >> d;
    for (int i = 0; i < n; ++i)
        connected[i] = i;
    for (int i = 0; i < m; ++i) {
        cin >> v1 >> v2;
        int a = getComponent(v1 - 1);
        int b = getComponent(v2 - 1);
        if (a != b)connected[a] = b;
    }
    for (int i = 0; i < d; ++i) {

        cin >> v1 >> v2 >> c;
        forBuild[i].v1 = v1 - 1;
        forBuild[i].v2 = v2 - 1;
        forBuild[i].cost = c;
    }
    sortBridge();
    answer = 0;
    for (int i = 0; i < d; ++i) {
        int a = getComponent(forBuild[i].v1);
        int b = getComponent(forBuild[i].v2);
        if (a != b) {
            connected[a] = b;
            answer += forBuild[i].cost;
        }
    }
    cout << answer;
          return 0;
}
```

Задание №5
Правильно одеть Кубик
ограничение по времени на тест: 2 *секунды*
ограничение по памяти на тест: *16 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Внешне Кубик выглядит как трехмерный куб.

Кубик считается одетым, если на каждой из его граней написано число. Кубик считается правильно одетым, если все числа на его гранях разные и ровно два из них являются простыми числами. У Кубика имеется последовательность из n натуральных чисел, которые он может написать на своих гранях. Определите сколько различных комбинаций правильно одеться имеется у Кубика. Две комбинации одевания считаются одинаковыми если одна может быть получена из другой вращением Кубика.

Входные данные:
Первая строка содержит целое число n ($0< n <10^5$) – количество чисел последовательности.
Следующая строка содержит по n натуральных чисел a_1, a_2, …, a_n ($1 \le a_i \le 10^5$).

Выходные данные:
Выведите искомое число.

Пример
входные данные
7
1 2 3 6 8 4 1

выходные данные
30

Решение на языке C++:

```
#include <iostream>
#include <set>
#include <cmath>

using namespace std;
bool simple(int n){
    if (n==1) return false;
    int a=int(sqrt(n));
    for (int i=2; i<=a; i++)
        if (n%i==0) return false;
    return true;
}
```

```
int main() {
        unsigned long long n,m;
        cin >> n;
        set<int> num,sim;
        for (int i = 0; i < n; ++i)
        {
                int k; cin >> k;
      if (simple(k))
        sim.insert(k);
      else
                num.insert(k);
        }
        n = num.size();
   m=sim.size();
        cout <<( n * (n - 1) * (n - 2) * (n - 3) * m * (m - 1) *5)/ 8;
        return 0;
}
```

Задание 6
Самая большая пирамидка

ограничение по времени на тест: *25 секунд*
ограничение по памяти на тест: *256 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубик складывает пирамидку из кругов. При построении пирамидки Кубик последовательно кладет один круг на другой большего радиуса. Все круги лежат на длинном прилавке в ряд. Кубик последовательно берет круги ряда и строит пирамидку. Каждый очередной круг Кубик может отбросить, а может попробовать разместить, если возможно, на пирамидке. Определите количество кругов в максимально высокой пирамидке, которую сможет построить Кубик.

Входные данные:
В первой строке задано целое число n ($1 \le n \le 10^4$) — количество кругов на прилавке.
Во второй строке заданы n целых чисел $d_1, d_2, \ldots, d_n$ — диаметры кругов ($d_i < 10^5$, $i = 1..n$).

Выходные данные:
Выведите искомое количество, являющегося решением задачи.

Пример
входные данные
12
9 6 4 2 2 1 8 5 4 3 2 1
выходные данные
7
Примечание
Максимальное количество кругов в пирамидке: (9 6 5 4 3 2 1)

Решение на языке C++:

```
#include <iostream>
#include <map>
#include <set>
using namespace std;

int main() {
	map<int, int> m;
	set<int> s;

	int n;
	cin >> n ;
	for (int i= 0; i < n; ++i) {
		int d;
```

```
		cin >> d;
		int t = 0;
		for (int x : s) {
			if (x > d && t < m[x]+1) t = m[x] + 1;
		}

		if (s.find(d) == s.end()) { m[d] = 1; s.insert(d); }
		if (m[d] < t) m[d] = t;
	}
	int res = 0;
	for (int x : s) {
		if (m[x] > res) res = m[x];
	}
	cout << res;
	return 0;
}
```

Задание 7

Красивые числа

ограничение по времени на тест :*15 секунд*
ограничение по памяти на тест: *32 мегабайт*
ввод: *стандартный ввод*
вывод: *стандартный вывод*

Кубик любит красивые числа. По мнению Кубика, число считается красивым, если оно представимо в виде $a^n b^m$ (a, b – простые числа, $n>0$, $m>0$) (Например, число *175* – красивое по мнению Кубика, так как представимо в виде $5^2 7^1$). Кубик хочет узнать сколько красивых чисел на отрезке $[x, y]$. Помогите Кубику решить эту задачу.

Входные данные:
Строка содержит *2* целых числа x, y ($1 \le x < y \le 2*10^{11}$).

Выходные данные:
Выведите число – ответ на задачу.

Пример
входные данные
20000 150000
выходные данные
41438

Решение на языке C++:

```
#include <iostream>
#include <cmath>

using namespace std;

int sim[41538];
int k = 0;
long long x, y;


bool simple(int n) {
	if (n == 1) return false;
	int a = round(sqrt(n));
	for (int i=0; i<k && sim[i]<=a;++i)
		if (n % sim[i] == 0) return false;
	return true;
}

long long count(int a, int b) {
	long long res = 0;
```

```
	for (long long stepa = a; stepa <= y; stepa = stepa * a)
		for (long long stepb = b; stepa * stepb <= y; stepb = stepb * b)
			if (stepa * stepb>= x) res++;
	return res;
}

int main() {

	for (int i= 2; i < 500000; ++i)
		if (simple(i)) sim[k++]=i;
	cin >> x >> y;
	long long result = 0;
	for (int i = 0; i < 41538; ++i)
		for (int j = i + 1; j < 41538; ++j)
			result = result + count(sim[i], sim[j]);
	cout << result;
	return 0;
}
```