A Квадрат

Нужно просто возвести целое число в квадрат.

Решение C++

```
#include<iostream>
using namespace std;
int main()
{
    long int n;
    cin>>n;
    cout<<n*n;
}
```

Решение Python

```
print(int(input())**2)
```

B Потепление

Нужно найти в массиве подотрезок из положительных чисел максимальной длины.

Решение C++

```
#include <iostream>

using namespace std;

int main() {
    int n;
    cin >> n;
    int res = 0, cur = 0;
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x;
        if (x > 0)
                cur++;
        else cur = 0;
        res = max(res, cur);
    }
    cout << res << endl;
}
```

Решение Python

```
n=input()
*k,=map(int,input().split())
w=0
m=0
for i in k:
    if i>0:
        m+=1
        w=max(m,w)
    else:
        m=0
print(w)
```

C Пароль

Можно вручную найти каждый ответ.

Решение C++

```cpp
#include <iostream>
using namespace std;
int main(){
	int n;
	cin>> n;
	int a[] = {0,8,98,986,9876,98764,987654,9876542,98765432,987654312};
	cout<< a[n];
	return 0;
}
```

Решение Python

```python
print([8, 98, 986, 9876, 98764, 987654, 9876542, 98765432, 987654312][int(input()) - 1])
```

D Пополнение гардероба

Ограничения задачи позволяют решать ее за $O(N^2)$ Мы можем для каждого элемента массива вложенным циклом проверить, встречалось ли данное число ранее.

Правильнее данную задачу решать с использованием структуры данных set или map. Таким образом асимптотика алгоритма будет O(N*logN)

Решение C++

```cpp
#include<iostream>
#include<set>
#include<vector>
using namespace std;
int main()
{
	set < int > s;
	int n;
	cin >> n;
	for (int i = 0; i < n; i++)
	{
		int a;
		cin >> a;
		if (s.find(a) == s.end())
		{
			s.insert(a);
			cout << "1 ";
		}
		else cout << "0 ";
	}

	return 0;
}
```

Решение Python

```
n=int(input())
ts=set()
for i in map(int,input().split()):
    if i not in ts:
        print(1,end=" ")
        ts.add(i)
    else:
        print(0,end=" ")
```

E Кастрюли

Если все размеры кастрюль различны, то мы можем вложить их все в одну «матрешку».

Сложности возникают, когда есть одинаковые. В принципе не сложно догадаться, что нужно посчитать количество одинаковых. Максимальное из этих чисел и будет ответом.

Можно использовать структуру данных map. Хотя ограничения задачи (10 000) позволяют использовать ассоциативный массив.

Решение C++

```
#include <bits/stdc++.h>
#define ll long long

using namespace std;
int n , d[10005];

int main()
{
    ios::sync_with_stdio(0);
    cin >> n;
    int ans = 0;
    for (int i=0;i<n;i++)
    {
        int x;
        cin >> x;
        d[x]++;
        ans = max(ans , d[x]);
    }
    cout << ans << endl;
    return 0;
}
```

Решение Python

```
n = int(input())
A = list(map(int, input().split()))
g = {}
for i in range(n):
    if A[i] not in g:
        g[A[i]] = 0
    g[A[i]] += 1
ans = 1
for x in g:
    ans = max(ans, g[x])
print(ans)
```

F Максимальня сумма на подотрезке

Будем поддерживать две переменные: ответ и текущая сумма чисел. Будем идти по массиву слева направо и добавлять новое число к текущей сумме. Если текущая сумма стала отрицательной, то обнуляем её. Если она больше ответа, то обновляем ответ.

Решение C++

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


int main() {

	int n;
	cin >> n;
	long long res = -1e9,sum = 0;

	for (int i = 0; i < n; i++){
		int v;
		cin >> v;
		sum += v;
		if (sum > res){
			res = sum;
		}
		if (sum < 0)
			sum = 0;
	}

	cout << res;

	return 0;
}
```

Решение Python

```
n = int(input())
a = list(map(int,input().split()))
sum = 0
best = -2147483647 - 1
for i in range(0,n):
    sum += a[i]
    best = max(sum,best)
    if(sum<0):
        sum=0
print(best)
```

Еще вариант решением динамическим программированием с использованием префиксных сумм. Как известно сумма чисел на отрезке от L до R равна pref[R] – pref[L-1], где pref[n] – сумма первых n чисел. Будем идти по массиву слева направо и считать максимальное значение на подотрезке, заканчивающимся на текущей позиции. Для этого надо хранить минимум среди всех префиксов слева.

G Переправа

Жадное решение. Метод двух указателей. Отсортируем всех по весу в порядке увеличения. Будем идти от самых легких.

К самому легкому будем пытаться добавить самого тяжелого. Если мы не можем этого сделать, то самый тяжелый ни с кем в паре уже точно поехать не сможет, поэтому везем его в одиночку и сдвигаем правый указатель от самого тяжелого влево. O(N)

Решение C++

```
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <string>
#include <unordered_set>
#include <cstring>
#include <unordered_map>
#include <algorithm>
#include <numeric>
#include <iomanip>
#include <cmath>
#include <queue>
#include <stack>
#include <fstream>
#include <utility>

typedef int64_t intt;

const intt MOD = 1000000007;
const intt inf = 1e18;

int main()
{
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(NULL);

    intt n, x; std::cin >> n >> x;
    std::vector<intt> v(n);
    for(int i = 0; i < n; i++) std::cin >> v[i];
    intt ans = 0;
    std::sort(v.begin(), v.end());
    intt i = 0, j = n-1;
    while(i <= j)
    {
        if(v[i] + v[j] <= x)
        {
            i++;
            j--;
        }
        else j--;
        ans++;
    }
    std::cout << ans;
}
```

Решение Python

```python
n, x = map(int, input().split())
P = [int(i) for i in input().split()]
i = 0
j = n - 1
num = 0
P.sort()
while i <= j:
    if P[i] + P[j] <= x:
        i += 1
    num += 1
    j -= 1
print(num)
```

H Ресторан

Это стандартная задача на поиск максимального количества пересекающихся отрезков на прямой. Отсортируем в одном массиве начала и концы отрезков с указанием, что это начало (+1) и конец (-1). Пройдем слева направо и будем добавлять к текущей сумме данные числа. Максимальное значение суммы и будет ответом. Это что-то похожее на проверку глубины вложенности скобочной последовательности.

Решение C++

```cpp
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define ar array

ar<int, 2> positions[400002];
int main() {
	int n;
	cin >> n;

	for(int i=0; i < 2*n; i+=2) {
		int a, b;
		cin >> a >> b;

		positions[i] = {a, 1};
		positions[i+1] = {b, -1};
	}

	sort(positions, positions + 2 * n);
	int mx = 0;
	int cur = 0;
	for(int i=0; i < 2 * n; i++) {
		cur += positions[i][1];
		mx = max(cur, mx);
	}

	cout << mx;
}
```

I Завод

Заметим, что мы можем легко посчитать количество деталей, которые можем произвести на всех станках за фиксированное время. Для этого просто пройдем по всем машинам и поделим это время на производительность каждой машины. Эти значения сложим (все машины могут работать параллельно). Также заметим, что если мы не сможем произвести нужно количество деталей за время T, то мы не сможем это сделать и за меньшее время. Это позволяет нам оптимизировать перебор ответа с помощью бинарного поиска.

По сути, это классическая задача на применение бинарного поиска по ответу.

Я рекомендую писать бинарный поиск таким образом, чтобы одна граница была всегда истиной (здесь правая), а другая всегда ложь. Цикл таким образом `while(l+1 < r){`

Решение C++

```
#include<bits/stdc++.h>
#define ll long long
#define st string
#define ld long double
#define ss second
#define ff first
#define ok define
using namespace std;
int main(){
    unsigned ll t, n, l = 0, r = 1e18, ans = 0, mid, x;
    cin >> n >> x;
    unsigned ll a[n];
    for(int i = 0; i < n; i++)
        cin >> a[i];
    while(l+1 < r){
        mid = (l + r) / 2;
        for(int i = 0; i < n; i++){
            ans += mid / a[i];
        }
        if(ans >= x)
            r = mid;
        else
            l = mid;
        ans = 0;
    }
    cout << r;
    return 0;
}
```

J Сумма двух чисел

Данная задача – пример эффективного использования структуры данных **map**.

Будем идти по массиву слева направо. Пусть нам нужно получить сумму x, а текущее число a. Тогда нужно быстро проверить, а было ли уже число x-a. Это позволяет делать map. Асимптотика O(N*logN)

Есть другой подход к решению с той же асимптотикой. Отсортируем массив. Воспользуемся методом двух указателей. Первым будем идти слева, а правый двигать влево в зависимости от суммы двух чисел и требуемого результата.

Решение C++

```
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define N 200000
using namespace std;

int main()
{
	int n,x;
	cin>>n>>x;

	map <int,int> mp;

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

		if (mp.find(x-a)!=mp.end())
		{
			cout<<mp[x-a]+1<<" "<<i+1;
			return 0;
		}
		mp[a]=i;
	}

	cout<<"-1";

    return 0;
}
```

K Найди дробь

Пусть x/y-требуемая несократимая дробь. (Дробь x/y является несократимой в том и только в том случае, если числа x и y взаимно простые, то есть их наибольший общий делитель равен 1) Нетрудно доказать, что x=НОК(a1, a2, ... , an), y=НОД(b1, b2, ... , bn), то есть число x совпадает с наименьшим общим кратным всех чисел a_i, а число y—с наибольшим общим делителем всех чисел b_i. В частности, если все числа a_i взаимно простые, то x=a1·a2·...·an. Поскольку все числа ai не превосходят 10^3, число n будет не более 10^{18}

В старых стандартах C++ есть встроенная функция __gcd, также в C++17 есть шаблонная реализация std::gcd

Также можно реализовать ее самостоятельно:

Алгоритм Евклида: на каждом шаге большее значение заменяем на остаток от деления большего на меньшее. Пока меньшее не станет равно нулю. Вроде как такая реализация работает быстрее, чем стандартные функции

Решение C++

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

long long gcd(long long x,long long y){
	return y == 0?x:gcd(y,x%y);
}

int main(){
	long long n,a,b;
	long long lcma,gcdb;
	cin>>n;
	for(int i = 0;i < n;i++){
		if(i == 0){
			cin>>lcma>>gcdb;
			continue;
		}
		cin>>a>>b;
		lcma = lcma/gcd(lcma,a)*a;
		gcdb = gcd(gcdb,b);
	}
	cout<<lcma<<" "<<gcdb<<endl;
}
```

Решение Python

```python
from math import gcd
def lcm(a, b): return a * b // gcd(a, b)

n = int(input())
c = 1
g = 0
for i in range(n):
    p, q = map(int, input().split())
    c = lcm(c, p)
    g = gcd(q, g)
print(c, g)
```

TO BE CONTINUED….

L Манхэттенский проект

Попробуйте сначала решить задачу для одномерного случая. Потом перейдите к двумерному. Если раскрыть модуль, то окажется, что можно упорядочить все точки по определенным характеристикам таким образом на каждый запрос нужно будет брать точку с минимальным/максимальным значением характеристики. Для двумерного случая это будет 4 массива, в одном точки с характеристикой -X1+X2 в другом -X1-X2, в третьем X1-X2, в четвертом X1+X2. Для 4-хмерного случая будет 16 массивов.

Решение C++

```
#include <bits/stdc++.h>

using namespace std;

int n, x[5], a, y[16];
multiset <int> s[16];

int main(){

    scanf ("%d", &n);
    for (int i = 1; i <= n; i++){
       scanf ("%d", &a);
       for (int j = 0; j < 4; j++)
              scanf ("%d", &x[j]);

         memset (y, 0, sizeof (y));
       for (int j = 0; j < 16; j++){
              for (int k = 0; k < 4; k++)
                     if (j & (1<<k))
                           y[j] += x[k];
                        else
                           y[j] -= x[k];
       }

       if (a == 1){
             for (int j = 0; j < 16; j++)
                    s[j].insert (y[j]);
       }
       else
            if (a == 2){
                  for (int j = 0; j < 16; j++)
                       s[j].erase (s[j].find (y[j]));
            }
             else{
                  int sol = 0;
                  for (int j = 0; j < 16; j++)
                         sol = max (sol, y[j] + *s[j ^ 15].rbegin());
                  printf("%d\n", sol);
             }
    }

      return 0;
}
```

M Мессенджер

Решение C++

```
#include <bits/stdc++.h>
using namespace std;

char s[262144];
char t[262144];
int n, sum = 0, mult = 1, len, num;

int main() {
	gets(s);
	len = strlen(s);
	scanf("%d", &n);
	for (int i = 0; i < n; ++i) {
		scanf("%d", &num);
		sum = sum + num * mult;
		mult *= -1;
		sum %= len;
	}
	if (sum < 0)
		sum += len;
	for (int i = 0; i < len; ++i)
		t[i] = s[(i + sum) % len];
	if (mult == 1) {
		puts(t);
	} else {
		for (int i = len - 1; i >= 0; --i) {
			putchar(t[i]);
		}
		putchar('\n');
	}
	return 0;
}
```

Решение C++ (декартово дерево)

```
#define _CRT_SECURE_NO_WARNINGS
#include <algorithm>
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
struct treap
{
	int sz;
	int pr;
	int val;
	bool rev;
	treap* l;
	treap* r;
	treap(int val) :l(NULL), r(NULL), pr(rand()), sz(1), rev(0), val(val)
{}
};
int get_sz(treap* t)
{
	if (t == NULL) return 0;
```

```
	return t->sz;
}
void upd(treap* &t)
{
	if (t == NULL) return;
	t->sz = get_sz(t->l) + get_sz(t->r) + 1;
}
void PUSH(treap* t)
{
	if (t && t->rev)
	{
		swap(t->l, t->r);
		if (t->l) t->l->rev ^= 1;
		if (t->r) t->r->rev ^= 1;
		t->rev = 0;
	}
}
void MERGE(treap* &t, treap* t1, treap* t2)
{
	PUSH(t1);
	PUSH(t2);
	if (!t1 || !t2)
	{
		t = t1 ? t1 : t2;
		return;
	}
	if (t1->pr < t2->pr)
	{
		MERGE(t2->l, t1, t2->l);
		t = t2;
	}
	else
	{
		MERGE(t1->r, t1->r, t2);
		t = t1;
	}
	upd(t);
}
void SPLIT(treap* t, treap* &t1, treap* &t2, int x, int add)
{
	PUSH(t);
	if (t == NULL)
	{
		t1 = t2 = NULL;
		return;
	}
	int key = add + get_sz(t->l) + 1;
	if (x < key)
	{
		SPLIT(t->l, t1, t->l, x, add);
		t2 = t;
	}
	else
	{
		SPLIT(t->r, t->r, t2, x, key);
		t1 = t;
	}
	upd(t1);
	upd(t2);
}
void INSERT(treap* &t, int x)
{
	MERGE(t, t, new treap(x));
}
```

```
treap* t;
void REVERSE(treap* &t, int x)
{
	treap *t1, *t2;
	SPLIT(t, t1, t2, x, 0);
	if (t1) t1->rev ^= 1;
	if (t2) t2->rev ^= 1;
	MERGE(t, t1, t2);
}
string s;
void PRINT(treap* t)
{
	if (t == NULL) return;
	PUSH(t);
	PRINT(t->l);
	cout << s[t->val-1];
	PRINT(t->r);
}
int main()
{
	getline(cin, s);
	int n, i;
	cin >> n;
	for (i = 1; i <= s.length(); i++) INSERT(t, i);
	for (i = 1; i <= n; i++)
	{
		int x;
		cin >> x;
		REVERSE(t, x);
	}
	PRINT(t);
	cout << endl;
	return 0;
}
```

Решение Python

```
s = str(input())
n = int(input())
k = 0
nn = 1
for i in range(n):
	x = int(input())
	k += x * nn
	nn *= -1
k = k % len(s)
s = s[k:] + s[:k]
if n % 2 == 1:
	s = s[::-1]
print(s)
```

N Жижа

Решение C++

```
#include <bits/stdc++.h>
#define fi first
#define se second
using namespace std;
using pii = pair<int, int>;
set<pii> st;
void insert(int l, int r)
{
	auto it = st.insert({r, l}).fi;
	if(it!=st.begin()&&(*(--it)).fi==l-1)
	{
		int pl = (*it).se;
		st.erase(it); st.erase({r, l});
		st.insert({r, pl});
	}
}
pii ins(int x, int p)
{
	auto it = st.lower_bound({x, 0});
	int l, r;
	if(it==st.end())
	{
		l = x, r = x + p - 1;
		insert(l, r);
	}
	else
	{
		if((*it).se>x)
		{
			if((*it).se>x+p)
			{
				l = x, r = x + p - 1;
				insert(l, r);
			}
			else
			{
				l = x, r = (*it).se - 1;
				int sr = (*it).fi;
				st.erase(it);
				insert(l, sr);
			}
		}
		else return ins((*it).fi+1, p);
	}
	return {l, r};
}
void del(int x)
{
	auto it = st.lower_bound({x, 0});
	if(it==st.end()||(*it).se>x) return;
	int l = (*it).se, r = (*it).fi;
	st.erase(it);
	if(x-1>=l) st.insert({x-1, l});
	if(r>=x+1) st.insert({r, x+1});
}
int main()
{
	int n; scanf("%d", &n);
	for(int i=1; i<=n; i++)
```

```
	{
		int x, p; scanf("%d", &x);
		if(x>0)
		{
			scanf("%d", &p);
			auto seg = ins(x, p);
			printf("%d %d\n", seg.fi, seg.se);
		}
		else del(-x);
	}
	return 0;
}
```