For practice session.txt

```
1. Проверить ONLINE_JUDGE  и LOCAL
2. Какой вердикт даёт throw
3. файлы или нет. Их имена
4. __int128
5. если codeblocks,то включить вывод контейнеров в настройках + -O2 + c++14(17)
```

Template.txt

```
//VISUAL ONLY:
//-----------------------------------
#define _CRT_SECURE_NO_WARNINGS
#pragma comment(linker, "/STACK:16777216")

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include<cassert>//assert
#include<vector>
#include<string>
#include<map>
#include<algorithm>
#include<deque>
#include<set>
#include<queue>
#include<stack>
#include<chrono>
#include <bitset>
#include <unordered_set>
#include <unordered_map>

//-------------------------------------

//GCC:
#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef unsigned long long ull;

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());
    //uniform_int_distribution<int>(0, i)(rnd) равномерное распределение
#ifndef ONLINE_JUDGE //МБ LOCAL
    freopen("in.txt", "rt", stdin);
    freopen("out.txt", "wt", stdout);
#endif


        return 0;
}
```

Python file in out.txt

```
f = open('workfile', 'w') #r read  w write  r+=r+w
f.readline()
for line in f:
        ...
f.write('asda')
```

1 Number theory

Extended euclid.txt

```
int gcd (int a, int b, int & x, int & y) {
        if (a == 0) {
                x = 0; y = 1;
                return b;
        }
        int x1, y1;
        int d = gcd (b%a, a, x1, y1);
        x = y1 - (b / a) * x1;
        y = x1;
        return d;
}
```

Modulo inverse.txt

```
//Решаем a*b=1 mod m относительно b
//Сводим к a*x+m*y=1 -> ax=1 mod m
//gcdex - extended euclid
int x, y;
int g = gcdex (a, m, x, y);
if (g != 1)
        cout << "no solution";
else {
        x = (x % m + m) % m;
        cout << x;
}
```

All modulo inverses.txt

```
//Для всех чисел [1,m-1] находим обратное по модулю m

r[1] = 1;
for (int i=2; i<m; ++i)
        r[i] = (m - (m/i) * r[m%i] % m) % m;
```

BigInt.txt

```
typedef vector<int> lnum;
const int base = 1000*1000*1000;

void print(lnum& a)
{
        printf ("%d", a.empty() ? 0 : a.back());
        for (int i=(int)a.size()-2; i>=0; --i)
                printf ("%09d", a[i]);
}

void read(lnum& a)
{
        sting s;
        cin>>s
        for (int i=(int)s.length(); i>0; i-=9)
        if (i < 9)
                a.push_back (atoi (s.substr (0, i).c_str()));
        else
                a.push_back (atoi (s.substr (i-9, 9).c_str()));
}
//a+=b
void add(lnum& a,lnum& b)
{
        int carry = 0;
        for (size_t i=0; i<max(a.size(),b.size()) || carry; ++i) {
                if (i == a.size())
                        a.push_back (0);
        a[i] += carry + (i < b.size() ? b[i] : 0);
```

```
        carry = a[i] >= base;
        if (carry)  a[i] -= base;
}


//a-=b
void sub(lnum& a,lnum& b)
{
        int carry = 0;
        for (size_t i=0; i<b.size() || carry; ++i) {
                a[i] -= carry + (i < b.size() ? b[i] : 0);
                carry = a[i] < 0;
                if (carry)  a[i] += base;
        }
        while (a.size() > 1 && a.back() == 0)
                a.pop_back();
}


// b<base
//a*=b
void mul(lnum& a,int b)
{
        int carry = 0;
        for (size_t i=0; i<a.size() || carry; ++i) {
                if (i == a.size())
                        a.push_back (0);
                long long cur = carry + a[i] * 1ll * b;
                a[i] = int (cur % base);
                carry = int (cur / base);
        }
        while (a.size() > 1 && a.back() == 0)
                a.pop_back();
}

//c=a/b
lnum div(lnum& a,lnum& b)
{
        lnum c (a.size()+b.size());
        for (size_t i=0; i<a.size(); ++i)
                for (int j=0, carry=0; j<(int)b.size() || carry; ++j) {
                        long long cur = c[i+j] + a[i] * 1ll * (j < (int)b.size() ? b[j] : 0) + carry;
                        c[i+j] = int (cur % base);
                        carry = int (cur / base);
                }
        while (c.size() > 1 && c.back() == 0)
                c.pop_back();
        return c;
}

//a/=b,  b<base
void div(lnum& a, int b)
{
        int carry = 0;
        for (int i=(int)a.size()-1; i>=0; --i) {
                long long cur = a[i] + carry * 1ll * base;
                a[i] = int (cur / b);
                carry = int (cur % b);
        }
        while (a.size() > 1 && a.back() == 0)
                a.pop_back();
}
```

Primes.txt

```
998244353, 10^4 + 7, 10^4 + 9, 10^6 + 3, 10^9 + 7, 10^9 + 9, 10^9 + 21,
10^9 + 33, 10^9 + 123, 10^15 + 159, 10^18 + 3, 10^18 + 31, 10^18 + 3111
```

Highly composite numbers.txt

```
d(840) = 32, d(9 240) = 64, d(83 160) = 128
d(720 720) = 240, d(8 648 640) = 448
d(91 891 800) = 768, d(931 170 240) = 1344
d(97 772 875 200) = 4032
d(963 761 198 400) = 6720
d(866 421 317 361 600) = 26880
d(897 612 484 786 617 600) = 103680
```

2 Data structures

Default segment tree.txt

```
int n, t[4*MAXN];
void build (int a[], int v, int tl, int tr) {  // вызвать с v=1 tl=0 tr=n-1
        if (tl == tr)
                t[v] = a[tl];
        else {
                int tm = (tl + tr) / 2;
                build (a, v*2, tl, tm);
                build (a, v*2+1, tm+1, tr);
                t[v] = t[v*2] + t[v*2+1];
        }
}

int sum (int v, int tl, int tr, int l, int r) {
        if (l > r)
                return 0;
        if (l == tl && r == tr)
                return t[v];
        int tm = (tl + tr) / 2;
        return sum (v*2, tl, tm, l, min(r,tm))
                + sum (v*2+1, tm+1, tr, max(l,tm+1), r);
}

void update (int v, int tl, int tr, int pos, int new_val) {
        if (tl == tr)
                t[v] = new_val;
        else {
                int tm = (tl + tr) / 2;
                if (pos <= tm)
                        update (v*2, tl, tm, pos, new_val);
                else
                        update (v*2+1, tm+1, tr, pos, new_val);
                t[v] = t[v*2] + t[v*2+1];
        }
}
```

Segment tree with range updates.txt

```
int n, t[4*MAXN];
void build (int a[], int v, int tl, int tr) {
        if (tl == tr)
                t[v] = a[tl];
        else {
                int tm = (tl + tr) / 2;
                build (a, v*2, tl, tm);
                build (a, v*2+1, tm+1, tr);
        }
}

void update (int v, int tl, int tr, int l, int r, int add) {
        if (l > r)
                return;
        if (l == tl && tr == r)
                t[v] += add;
```

```
        else {
                int tm = (tl + tr) / 2;
                update (v*2, tl, tm, l, min(r,tm), add);
                update (v*2+1, tm+1, tr, max(l,tm+1), r, add);
        }
}

int get (int v, int tl, int tr, int pos) {
        if (tl == tr)
                return t[v];
        int tm = (tl + tr) / 2;
        if (pos <= tm)
                return t[v] + get (v*2, tl, tm, pos);
        else
                return t[v] + get (v*2+1, tm+1, tr, pos);
}
```

DSU.txt

```
void make_set (int v) {
        parent[v] = v;
        rank[v] = 0;
}

int find_set (int v) {
        if (v == parent[v])
                return v;
        return parent[v] = find_set (parent[v]);
}

void union_sets (int a, int b) {
        a = find_set (a);
        b = find_set (b);
        if (a != b) {
                if (rank[a] < rank[b])
                        swap (a, b);
                parent[b] = a;
                if (rank[a] == rank[b])
                        ++rank[a];
        }
}
```

Treap.txt

```
struct tree_node
{
        tree_node *left, *right;
        int key, priority;
        tree_node(int new_key)
        {
                key = new_key;
                priority = rand();
                left = right = nullptr;
        }
};

pair<tree_node*, tree_node*> split(tree_node *root, int key)
{
        if (root == nullptr)
                return make_pair(nullptr, nullptr);
        if (root->key < key)
        {
                pair<tree_node*, tree_node*> splitted = split(root->right, key);
                root->right = splitted.first;
                return make_pair(root, splitted.second);
        }
        else
        {
```

```
                pair<tree_node*, tree_node*> splitted = split(root->left, key);
                root->left = splitted.second;
                return make_pair(splitted.first, root);
        }
}

tree_node* merge(tree_node *left, tree_node *right)
{
        if (left == nullptr || right == nullptr)
                return right == nullptr ? left : right;
        if (left->priority > right->priority)
        {
                left->right = merge(left->right, right);
                return left;
        }
        else
        {
                right->left = merge(left, right->left);
                return right;
        }
}

void insert(tree_node &*root, int key)
{
        pair<tree_node*, tree_node*> splitted = split(root, key);
        merge(merge(splitted.first, new node(key)), splitted.second);
}

void erase(tree_node &*root, int key)
{
        tree_node *left = nullptr, *middle = nullptr, *right = nullptr;
        split(root, left, middle, key);
        split(middle, middle, right, key + 1);
        merge(middle, middle->left, middle->right);
        merge(root, left, middle);
        merge(root, root, right);
}
```

3 Geometry

Geometry basics.txt

```
struct Point
{
    double x,y;
    Point(){}
    Point(double _x,double _y)
    {
        x=_x;
        y=_y;
    }
};

istream& operator>>(istream& in,Point& p)
{
    in>>p.x;
    in>>p.y;
    return in;
}

ostream& operator<<(ostream& out,const Point& p)
{
    out<<p.x<<" "<<p.y<<'\n';
    return out;
}
```

```
Point makeVector(Point& p1, Point& p2)
{
    return Point(p2.x - p1.x, p2.y - p1.y);
}

double operator%(const Point& p1,const Point& p2)
{
    return p1.x*p2.x+p1.y*p2.y;
}

Point operator+(Point p1,Point p2)
{
    return Point(p1.x+p2.x,p1.y+p2.y);
}

Point operator-(Point p)
{
    return Point(-p.x,-p.y);
}

Point operator-(Point p1,Point p2)
{
    return p1+(-p2);
}

Point operator*(Point p,double x)
{
    return Point(p.x*x,p.y*x);
}

Point operator*(double x,const Point p)
{
    return p*x;
}

double operator*(const Point p1,const Point p2)
{
    return p1.x*p2.y-p2.x*p1.y;
}

double lenSq(const Point& p)
{
    return p%p;
}

double len(const Point& p)
{
    return sqrt(lenSq(p));
}

int sign(double x)
{
    if (x<0) return -1;
    if(x>0) return 1;
    return 0;
}

Point rotate(const Point& p)
{
    return Point(-p.y, p.x);
}

Point rotate(const Point& p,double cosa, double sina)
{
        Point v = p;
        Point u = rotate(v);
```

```
        Point w = v * cosa + u * sina;
        return w;
}
```

Convex polygon area.txt

```
double area(vector<Point>& figure)
{
    double sq = 0;
    for (int i = 0; i < figure.size() - 1; ++i)
        sq += (figure[i + 1].x - figure[i].x)*(figure[i].y + figure[i + 1].y) / 2.;
    sq+=(figure[0].x-figure[figure.size()-1].x)*(figure[figure.size()-1].y+figure[0].y)/2.;
    return sq;
}
```

Convex hull.txt

```
vector<Point>convexHull(vector<Point> points)
{
    Point pStart = points[0];
    int ind = 0;
    for (int i = 0; i < points.size(); ++i)
    {
        if (pStart.y > points[i].y || (pStart.y == points[i].y && pStart.x > points[i].x))
        {
            pStart = points[i];
            ind = i;
        }
    }
    swap(points[0], points[ind]);
    sort(points.begin() + 1, points.end(), [&pStart](Point p1, Point p2) {
        Point v1 = makeVector(pStart, p1);
        Point v2 = makeVector(pStart, p2);
        int vp = v1*v2;
        return vp > 0 || (vp == 0 && len(v1) < len(v2));
    });

    vector<Point>hull(points.size() + 1);
    int top = 1;
    hull[0] = (points[0]);
    hull[1] = (points[1]);

    for (int i = 2; i < points.size(); ++i)
    {
        while (top >= 1 && (makeVector(hull[top], hull[top - 1])*makeVector(points[i], hull[top]) <= 0))
        {
            --top;
        }
        ++top;
        hull[top] = points[i];
    }

    return vector<Point>(hull.begin(), hull.begin() + top + 1);
}
```

4 Graph

Dijkstra.txt

```
const int INF = 1000000000;

int main() {
        int n;

        vector < vector < pair<int,int> > > g (n);

        int s = 0; // стартовая вершина
```

```
        vector<int> d (n, INF),  p (n);
        d[s] = 0;
        priority_queue < pair<int,int> > q;
        q.push (make_pair (0, s));
        while (!q.empty()) {
                int v = q.top().second,  cur_d = -q.top().first;
                q.pop();
                if (cur_d > d[v])  continue;

                for (size_t j=0; j<g[v].size(); ++j) {
                        int to = g[v][j].first,
                                len = g[v][j].second;
                        if (d[v] + len < d[to]) {
                                d[to] = d[v] + len;
                                p[to] = v;
                                q.push (make_pair (-d[to], to));
                        }
                }
        }
}
```

Floyd-Warshall.txt

```
//d[n][n]
for (int k=0; k<n; ++k)
        for (int i=0; i<n; ++i)
                for (int j=0; j<n; ++j)
                        if (d[i][k] < INF && d[k][j] < INF) //на случай ребер отрицательного веса
                                d[i][j] = min (d[i][j], d[i][k] + d[k][j]);
```

Ford-Bellman.txt

```
//Граф задан списком ребер struct {int a,b,cost;}
void solve() {
        vector<int> d (n, INF);
        d[v] = 0;
        for (;;) {
                bool any = false;
                for (int j=0; j<m; ++j)
                        if (d[e[j].a] < INF)
                                if (d[e[j].b] > d[e[j].a] + e[j].cost) {
                                        d[e[j].b] = d[e[j].a] + e[j].cost;
                                        any = true;
                                }
                if (!any)  break;
        }
}
```

Strongly connected components.txt

```
//ЛЕНЬ ТЕСТИТЬ, МБ БАГИ!
vector<int>order;
vector<int>used;

void dfs1(vector<vector<int> >& g, int start)
{
    used[start] = 1;
    for (int i = 0; i < g[start].size(); ++i)
    {
        int v = g[start][i];
        if (!used[v])
        {
            dfs1(g, v);
        }
    }
    order.push_back(start);
}
```

```
//gr - reversed graph
void dfs2(vector<vector<int> >& gr, vector<int>& comp, int start, int curCompN)
{
    used[start] = curCompN;
    comp.push_back(start);
    for (int i = 0; i < gr[start].size(); ++i)
    {
        int v = gr[start][i];
        if (!used[v])
        {
            dfs2(gr, comp, v, curCompN);
        }
    }
}
used.assign(n, false);
for (int i = 0; i < n; ++i)
{
    if (!used[i])
        dfs1(g, i);
}
used.assign(n, false);
vector<vector<int> >componets;
int x = 1;
for (int i = 0; i < n; ++i)
{
    int v = order[n - i - 1];
    if (!used[v])
    {
        vector<int>comp;
        dfs2(gr, comp, v, x);
        componets.push_back(comp);
        ++x;
    }
}

vector<vector<int> >condGraph(componets.size());
//на этом этапе в used находятся номера компонент связность
for (int i = 0; i < g.size(); ++i)
{
    for (int j = 0; j < g[i].size(); ++j)
    {
        if (used[i] != used[g[i][j]])
            condGraph[used[i] - 1].push_back(used[g[i][j]] - 1);
    }
}

/* Это если в списках смежность не нужны повторения
for (int i = 0; i < condGraph.size(); ++i)
{
    set<int>s(condGraph[i].begin(), condGraph[i].end());
    condGraph[i].assign(s.begin(), s.end());
}
*/
```

5 Game Theory

Game theory approach.txt

```
Применение теоремы Шпрага-Гранди
Опишем наконец целостный алгоритм, применимый
к любой равноправной игре двух игроков для определения
 выигрышности/проигрышности текущего состояния v.

Функция, которая каждому состоянию игры ставит в
соответствие ним-число, называется функцией Шпрага-Гранди.
```

```
Итак, чтобы посчитать функцию Шпрага-Гранди для текущего
состояния некоторой игры, нужно:

Выписать все возможные переходы из текущего состояния.
Каждый такой переход может вести либо в одну игру,
либо в сумму независимых игр.
В первом случае - просто посчитаем функцию Гранди
рекурсивно для этого нового состояния.

Во втором случае, когда переход из текущего состояния приводит
в сумму нескольких независимых игр -
рекурсивно посчитаем для каждой из этих игр функцию Гранди,
а затем скажем, что функция Гранди суммы игр равна XOR-сумме значений этих игр.

После того, как мы посчитали функцию Гранди
для каждого возможного перехода - считаем mex от этих значений,
и найденное число - и есть искомое значение Гранди для текущего состояния.
Если полученное значение Гранди равно нулю,
то текущее состояние проигрышно, иначе - выигрышно.

функция mex от множества чисел возвращает наименьшее неотрицательное число,
 не встречающееся в этом множестве
```

6 Strings

Prefix function.txt

```
int[] prefixFunction(string s):
  p[0] = 0
  for i = 1 to s.length - 1
      k = p[i - 1]
      while k > 0 and s[i] != s[k]
          k = p[k - 1]
      if s[i] == s[k]
          k++
      p[i] = k
  return p
```

Manaker.txt

```
int[] calculate1(string s):
  int l = 0
  int r = -1
  for i = 1 to n
    int k = 0
    if i <= r
       k = min(r - i, d[r - i + l])
    while i + k + 1 <= n and i - k - 1 > 0 and s[i + k + 1] == s[i - k - 1]
       k++
     d1[i] = k
     if i + k > r
       l = i - k
       r = i + k
  return d1

int[] calculate2(string s):
        int l = 0
        int r = -1
        for i = 1 to n
        int k = 0
        if i <= r
           k = min(r - i + 1, d[r - i + l + 1])
        while i + k <= n and i - k - 1 > 0 and s[i + k] == s[i - k - 1]
           k++
         d2[i] = k
         if i + k - 1 > r
```

```
            l = i - k
            r = i + k - 1
        return d2
```

Suffix automata.txt

```
struct state {
        int len, link;
        map<char,int> next;
};
const int MAXLEN = 100000;
state st[MAXLEN*2];
int sz, last;

void sa_init() {
        sz = last = 0;
        st[0].len = 0;
        st[0].link = -1;
        ++sz;
        /*
        // этот код нужен, только если автомат строится много раз для разных строк:
        for (int i=0; i<MAXLEN*2; ++i)
                st[i].next.clear();
        */
}
void sa_extend (char c) {
        int cur = sz++;
        st[cur].len = st[last].len + 1;
        int p;
        for (p=last; p!=-1 && !st[p].next.count(c); p=st[p].link)
                st[p].next[c] = cur;
        if (p == -1)
                st[cur].link = 0;
        else {
                int q = st[p].next[c];
                if (st[p].len + 1 == st[q].len)
                        st[cur].link = q;
                else {
                        int clone = sz++;
                        st[clone].len = st[p].len + 1;
                        st[clone].next = st[q].next;
                        st[clone].link = st[q].link;
                        for (; p!=-1 && st[p].next[c]==q; p=st[p].link)
                                st[p].next[c] = clone;
                        st[q].link = st[cur].link = clone;
                }
        }
        last = cur;
}
```

7 Other

ternary search.txt

```
//f возрастает, потом убывает. Или наоборот
double l = ..., r = ..., EPS = ...; // входные данные
while (r - l > EPS) {
   double m1 = l + (r - l) / 3,
      m2 = r - (r - l) / 3;
   if (f (m1) < f (m2))
      l = m1;
   else
      r = m2;
}
```

	Comment	M	C	A
A				
B				
C				
D				
E				
F				
G				
H				
I				
J				
K				
L				
M				