# *Notebook*

*vici*

October 1, 2013

# Contents

# 1 Basic

## 1.1 Header

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <cmath>
#include <string>
#include <vector>
#include <queue>
#include <set>
#include <map>
#include <ctime>

#define inf 0x3f3f3f3f
#define Inf 0x3FFFFFFFFFFFFFFFLL

using namespace std;

int main()
{

    return 0;
}
```

## 1.2 KeyWords

```
//set 1
foreach string __int64 int64 map set vector bitset
queue deque priority_queue node ll iterator pii vii
 multiset ull u64

//set 2
__builtin_popcount distance pow abs min max sqrt sin
cos acos asin atan2 inf Inf root null NULL maxn maxm
MAXN MAXM pi eps EPS
```

## 1.3 Macros and Templates

```
#define debug(X) cerr<<""#X":␣"<<(X)<<"\n"
template<class T> void pv(T a, T b) {
    for (T i = a; i != b; ++i) cout << *i << "␣"; cout << endl;
}
template<class T> void chmin(T &t, T f) { if (t > f) t = f; }
template<class T> void chmax(T &t, T f) { if (t < f) t = f; }
```

## 1.4 Faster_IO(G++ is better)

```
inline int getint() {
    int ret = 0;
    char c;
    while (!isdigit(c = getchar()));
    do ret = (ret << 3) + (ret << 1) + c - '0';
        while (isdigit(c = getchar()));
    return ret;
}

inline int nextInt() {
    char c = getchar();
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    int ret = 0, neg = 0; if (c == '-') neg = 1, c = getchar();
    do ret = (ret << 3) + (ret << 1) + c - '0';
        while (isdigit(c = getchar()));
    return neg ? -ret : ret;
}
int A[20], k;
inline void printInt(int x) {
    if (x < 0) putchar('-'), x = -x;
    else if (x == 0) { putchar('0'); return; }
    k = 0; while (x) A[k++] = x % 10, x /= 10;
    for (int i = k - 1; ~i; i--) putchar('0' + A[i]);
    putchar('\n');
}
inline double nextDouble() {
    char c; int i = 1;
    double ret = 0;
    do c = getchar();
        while (c != '-' && c != '.' && (c < '0' || c > '9'));
    bool neg = false;
    if (c == '-') neg = true, c = getchar();
    do ret = ret * 10 + c - '0';
        while (isdigit(c = getchar()));
    if(c != '.') return neg ? -ret : ret;
    c = getchar();
    do ret += (c - '0') / pow(10.0, i++);
        while (isdigit(c = getchar()));
    return neg ? -ret : ret;
}
```

## 1.5 Notes

```
double pi = 3.14159265358...; // no 'f' appended
#define maxn 200 + 20 // it's not safe
int const mod = 1000000007; // use "const" to accelerate
judge ``mp.find(x) != mp.end()"␣is␣better␣than␣``ret␣+=␣mp[x]" directly

0x3f3f3f3f  =  1061109567 (recommended)
0x7f7f7f7f  =  2139062143
0x3FFFFFFFFFFFFFFFLL  =  4611686018427387903  (recommended)
0x7FFFFFFFFFFFFFFFLL  =  9223372036854775807
```

```
//Increase the Stack Size(Only C++)
#pragma comment(linker, "/STACK:36777216")
//memset
memset(dpMin, 0x3f, sizeof(dpMin)); // inf
memset(dpMax, 0xc0, sizeof(dpMax)); // -inf
//for_bit
for (int i = a ; i != 0 ; i = (i - 1) & a)
//count bit
static int countbit[1024];
for (int i = 1; i < 1024; ++i) countbit[i] = 1 + countbit[i - ((i ^ (i -
     1)) & i)];
//sort by lexicographic
int cmp(const void *a, const void *b) {
    char *x = (char *)a;
    char *y = (char *)b;
    return strcmp(x, y);
}
qsort(str, n, sizeof(str[0]), cmp);
void RE(){int *PP=NULL;*PP=1;}
void MLE(){int *a=new int[100000000];exit(0);}
void TLE(){for(;;);}
```

# 2 Maths

## 2.1 gcd

```
int gcd(int a, int b) {
    if(b == 0) return a;
    else return gcd(b, a % b);
}
```

## 2.2 fgcd

```
#define eps 1e-8
double fgcd(double a, double b) {
    if(b > -eps && b < eps) {
        return a;
    } else {
        return fgcd( b, fmod(a, b) );
    }
}
```

## 2.3 sieve

```
//cnt[1e7] = 664579, cnt[1e6] = 78498
//mark[i] : the minimum factor of i (when i is a prime, mark[i] == i)
int const maxn = 1e7;
int pri[maxn], mark[maxn], cnt;
void sieve() {
    cnt = 0, mark[0] = mark[1] = 1;
    for (int i = 2; i < maxn; i++) {
        if (!mark[i]) pri[cnt++] = mark[i] = i;
        for (int j = 0; pri[j] * i < maxn; j++) {
            mark[ i * pri[j] ] = pri[j];
            if (i % pri[j] == 0) break;
        }
    }
}
```

## 2.4 sieve the number of divisors (O(nlogn))

```
int const maxn = 1e6;
int nod[maxn];
void __sieve_nod() {
    for (int i = 1; i < maxn; ++i) {
        for (int j = i; j < maxn; j += i) {
            ++nod[j];
        }
    }
}
```

## 2.5 sieve the number of divisors (O(n))

```
int const maxn = 1e6;
int pri[maxn], e[maxn], divs[maxn], cnt;
void __sieve_nod() {
    cnt = 0;
    divs[0] = divs[1] = 1;
    for (int i = 2; i < maxn; ++i) {
          if (!divs[i]) {
           divs[i] = 2;
           e[i] = 1;
           pri[cnt++] = i;
        }
        for (int j = 0; i * pri[j] < maxn; ++j) {
            int k = i * pri[j];
            if (i % pri[j] == 0) {
                e[k] = e[i] + 1;
                divs[k] = divs[i] / (e[i] + 1) * (e[i] + 2);
                break;
            }
            else {
                e[k] = 1;
                divs[k] = divs[i] << 1;
            }
        }
    }
}
```

## 2.6 sieve phi

```
int const maxn = 1e6;
int pri[maxn], cnt;
int phi[maxn];
void __sieve_phi() {
    cnt = 0, phi[1] = 1;
    for (int i = 2; i < maxn; ++i) {
        if (!phi[i]) {
            pri[cnt++] = i;
            phi[i] = i - 1;
        }
        for (int j = 0; pri[j] * i < maxn; ++j) {
            if (!(i % pri[j])) {
                phi[i * pri[j]] = phi[i] * pri[j];
                break;
            }
            else {
                phi[i * pri[j]] = phi[i] * (pri[j] - 1);
            }
        }
    }
}
```

## 2.7 sieve 100000 primes > 1e12

```
//or java.BigInteger -> nextProbablePrime();
typedef long long ll;
int const maxn = 4e6;
ll const start = 1e12, end = start + 3e6;

int pri[maxn], cnt; bool mark[maxn];
ll pl[maxn]; int pnt; bool markl[maxn];

void __sieve_large() {
    cnt = 0, mark[0] = mark[1] = true;
    for (int i = 2; i < maxn; ++i) {
        if (!mark[i]) pri[cnt++] = i;
        for (int j = 0; i * pri[j] < maxn; ++j) {
            mark[i * pri[j]] = true;
            if (!(i % pri[j])) break;
        }
    }
    ll pos;
    for (int i = 0; i < cnt; ++i) {
        if (start % pri[i] == 0) pos = start;
        else pos = start - start % pri[i] + pri[i];
        for (; pos <= end; pos += pri[i]) {
            markl[pos - start] = true;
        }
    }
    pnt = 0;
    for (int i = 0; i <= end - start; ++i) {
        if (!markl[i]) pl[pnt++] = start + i;
    }
}
```

## 2.8 powMod

```
typedef long long ll;
ll powMod(ll a, ll b, ll c){
     ll res = 1LL;
    while (b) {
        if(b & 1) res = res * a % c;
        a = a * a % c;
        b >>= 1;
    }
    return res;
}
```

## 2.9 powMod_plus

```
typedef long long ll;
inline ll mulMod(ll a, ll b, ll c){
    ll res = 0LL;
    for (; b; b >>= 1, a = (a << 1) % c ) {
        if (b & 1) res = (res + a) % c;
    }
    return res;
}
ll powMod(ll a, ll b, ll c){
    ll res = 1LL;
    for (; b; b >>= 1, a = mulMod(a, a, c) ) {
        if (b & 1) res = mulMod(res, a, c);
    }
    return res;
}
```

## 2.10 Miller-Rabin

```
bool suspect(ll a, int s, ll d, ll n) {
    ll x = powMod(a, d, n);
    if (x == 1) return true;
    for (int r = 0; r < s; ++r) {
        if (x == n - 1) return true;
        x = x * x % n;
    } return false;
}
// {2, 7, 61, -1} is for n < 4759123141 (= 2^32)
// {2, 3, 5, 7, 11, 13, 17, 19, 23, -1} is for n < 10^16 (at least)
bool isPrime(ll n) {
    if (n <= 1 || (n > 2 && n % 2 == 0)) return false;
    int test[] = {2,3,5,7,11,13,17,19,23,-1};
    ll d = n - 1, s = 0;
    while (d % 2 == 0) ++s, d /= 2;
    for (int i = 0; test[i] < n && test[i] != -1; ++i)
        if (!suspect(test[i], s, d, n)) return false;
    return true;
}
```

## 2.11 Pollard-Rho

```
ll pollard_rho(ll n, ll c){ // c can be (rand() % n)
    ll d, x = rand() % n, y = x;
    for(ll i=1,k=2; ;i++){
        x = (mulMod(x, x, n) + c) % n;
        d = gcd(y - x, n);
        if (d > 1 && d < n) return d;
        if (x == y) return n;
        if (i == k) y = x, k += k;
    }
    return 0;
}
```

## 2.12 find_factors

```
int const maxf = 500;
int facs[maxf];
int DecFun(int n) {
```

```
    int cnt = 0;
    for(int i = 2; i * i <= n; i += 2) {
        while(!(n % i) ) {
            n /= i;
            facs[cnt++] = i;
        }
        if(i == 2) i--;
    }
    if(n > 1) facs[cnt++] = n;
    return cnt;
}
```

## 2.13 find_factors_plus

```
// sieve() first & (n < maxn)
int const maxf = 500;
int facs[maxf];
int __find_factors(int n)
{
    int cnt = 0;
    while (mark[n] != 1)
    {
        facs[cnt++] = mark[n];
        n /= mark[n];
    }
    return cnt;
}
```

## 2.14 phi

```
int phi(int n) {
    int ret = n;
    for (int i = 2; i * i <= n; i += 2) {
        if(n % i == 0) {
            ret = ret / i * (i - 1);
            while(n % i == 0) n /= i;
        }
        if(i == 2) i--;
    }
    if (n > 1) ret = ret / n * (n - 1);
    return ret;
}
```

## 2.15 phi_plus

```
// sieve() first & (n < maxn)
int phi(int n)
{
    int ret = n, t;
    while ((t = mark[n]) != 1)
    {
        ret = ret / t * (t - 1);
        while (mark[n] == t) n /= mark[n];
    }
    return ret;
}
```

## 2.16 Dfun (the number of divisors (from 1 to n))

```
// for n < 1e8 the maximum Dfun(n) is Dfun(720720) = 240
int DFun(int n) {
    int res = 1, t;
    for(int i = 2; i * i <= n; i += 2) {
        if(!(n % i) ) {
            t = 1;
            while(n % i == 0){
                t++, n /= i;
            }
            res = res * t;
        } if(i == 2) i--;
    }
    if(n > 1) res *= 2;
    return res;
}
```

## 2.17 Dsum (the sum of all divisors (from 1 to n))

```
int DsFun(int n) {
    int res = 1, m = n, t;
    for(int i = 2; i * i <= n; i += 2) {
        if(!(n % i) ) {
            t = i * i, n /= i;
            while(n % i == 0) {
                t *= i, n /= i;
            }
            res = res * (t - 1) / (i - 1);
        }
        if(i == 2) i--;
    }
    if(n > 1) res *= n + 1;
    return res;
}
```

## 2.18 C(n,m) (dp)

```
int const maxn = 30;
int C[maxn][maxn];
void Cinit() {
   for (int i = 0; i < maxn; ++i) {
      C[i][0] = 1;
      for (int j = 1; j <= i; ++j) {
         C[i][j] = C[i - 1][j - 1] + C[i - 1][j];
```

```
    }
  }
}
```

## 2.19 C(n,m)%mod (div)

```
typedef long long ll;
int const maxn = 1000100;
int const maxm = 100100; //cnt ~ maxn / 10
ll const mod = 1000000007;
int pri[maxm], cnt; bool mark[maxn];
int p1[maxm], p2[maxm], p3[maxm];

void sieve() {
    cnt = 0, mark[0] = mark[1] = true;
    for (int i = 2; i < maxn; ++i) {
        if (!mark[i]) pri[cnt++] = i;
        for (int j = 0; i * pri[j] < maxn; ++j) {
            mark[i * pri[j]] = true;
            if (!(i % pri[j])) break;
        }
    }
}

int div(int *p, int n) {
    for (int i = 0, t; ; ++i) {
        if (pri[i] > n) return i;
        for (p[i] = 0, t = n; t; t /= pri[i]) {
            p[i] += t / pri[i];
        }
    }
}

ll C(int a, int b) { // a >= b, sieve() first!
    int l1 = div(p1, a);
    int l2 = div(p2, a - b);
    int l3 = div(p3, b);
    ll ret = 1LL;
    for (int i = 0; i < l1; ++i) {
        if (i < l2) p1[i] -= p2[i];
        if (i < l3) p1[i] -= p3[i];
        if (p1[i]) {
            ll r = 1LL, t = pri[i];
            while (p1[i]) {
                if (p1[i] & 1) r = r * t % mod;
                t = t * t % mod;
                p1[i] >>= 1;
            }
            ret = ret * r % mod;
        }
    }
    return ret;
}
```

## 2.20 C(n,m)%mod (inv)

```
\\mod must be a prime
typedef long long ll;
ll const mod = 1000000007;
int const maxn = 100100;
ll fac[maxn], inv[maxn];
ll C(int n, int m) {
    return fac[n] * inv[m] % mod * inv[n - m] % mod;
}
ll powMod(ll a, ll b) {
    ll ret = 1LL;
    while (b) {
        if (b & 1) ret = ret * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ret;
}
void Cinit() {
    fac[0] = inv[0] = 1LL;
    for (int i = 1; i < maxn; ++i) {
        fac[i] = fac[i - 1] * i % mod;
        inv[i] = powMod(fac[i], mod - 2);
    }
}
```

## 2.21 Nim_mul

```
int data[4][4]={{0,0,0,0},{0,1,2,3},{0,2,3,1},{0,3,1,2}};
int md[MAXM]={2,4,16,256,65536};
int nim_mul(int x,int y){
    if(x<y)return nim_mul(y,x);
    if(x<4)return data[x][y];
    int a,M,p,q,s,t,c1,c2,c3;
    for(a=0;a<5;a++){
        if(md[a]>x)break;
    }
    a--;
    M=md[a];
    p=x/M;q=x%M;
    s=y/M;t=y%M;
    c1=nim_mul(p,s);
    c2=nim_mul(p,t)^nim_mul(q,s);
    c3=nim_mul(q,t);
    return M*(c1^c2)^c3^nim_mul(M/2,c1);
}
```

## 2.22 MLES

```
typedef __int64 ll;
ll Extended_Euclid(ll a,ll b,ll *x,ll *y){
    if(b==0){
            *x=1;
            *y=0;
            return a;
    }
    ll d=Extended_Euclid(b,a%b,x,y);
    ll t=*x;
    *x=*y;
    *y=t-a/b*(*y);
    return d;
}
ll MLES(ll a,ll b,ll n){
    ll d,x,y;
    d=Extended_Euclid(a,n,&x,&y);
    ll x0;
    if(b%d==0){
        x0=(x*b/d)%n+n;
    }
    else return -1;
    return x0%(n/d);
}
```

## 2.23 Place $n$ Balls into $m$ Boxes

| Balls | Boxes | Empty Boxes | Answer |
|---|---|---|---|
| Different | Different | Yes | $m^n$ |
| Different | Different | No | $m!S(n,m)$ |
| Different | Same | Yes | $S(n,1)+S(n,2)+\ldots+S(n,\min(n,m))$ |
| Different | Same | No | $S(n,m)$ |
| Same | Different | Yes | $C(n+m-1,n)$ |
| Same | Different | No | $C(n-1,m-1)$ |
| Same | Same | Yes | $F(n,m)$ |
| Same | Same | No | $F(n-m,m)$ |

### 2.23.1 Inits

```
#define maxn 55
typedef long long ll;
// + mod
ll C[maxn + 1][maxn + 1];
void Cinit() {
    for (int i = 0; i <= maxn; ++i) {
        C[i][0] = 1LL;
        for (int j = i; j >= 1; --j) {
            C[i][j] = C[i - 1][j] + C[i - 1][j - 1];
        }
    }
}
```

```
ll S[maxn + 1][maxn + 1]; //Strling2[]
void Sinit() {
    S[0][0] = 1;
    for (int i = 1; i <= maxn; ++i) {
        S[i][1] = 1;
        for (int j = 2; j <= maxn; ++j) {
            S[i][j] = S[i - 1][j - 1] + j * S[i - 1][j];
        }
    }
}
ll F[maxn + 1][maxn + 1];
void Finit() {
    for (int i = 0; i <= maxn; ++i) F[i][1] = F[0][i] = 1;
    for (int i = 1; i <= maxn; ++i) {
        for (int j = 2; j <= maxn; ++j) {
            F[i][j] = F[i][j - 1];
            if (i >= j) F[i][j] += F[i - j][j];
        }
    }
}

// A Special Edition For the Memory Limit
ll const mod = 1000000007;
ll f[maxn];
int dp(int n, int m) {
    if (n > m) n = m;
    for (int i = 0; i <= m; ++i) f[i] = 1LL;
    for (int j = 2; j <= n; ++j) {
        for (int i = j; i <= m; ++i) {
            f[i] += f[i - j];
            if (f[i] >= mod) f[i] -= mod;
        }
    }
}
```

### 2.23.2 $C(n,m)$

| n \ m | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | | | | | | |
| 1 | 1 | 1 | | | | | | | | | |
| 2 | 1 | 2 | 1 | | | | | | | | |
| 3 | 1 | 3 | 3 | 1 | | | | | | | |
| 4 | 1 | 4 | 6 | 4 | 1 | | | | | | |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 | | | | | |
| 6 | 1 | 6 | 15 | 20 | 15 | 6 | 1 | | | | |
| 7 | 1 | 7 | 21 | 35 | 35 | 21 | 7 | 1 | | | |
| 8 | 1 | 8 | 28 | 56 | 70 | 56 | 28 | 8 | 1 | | |
| 9 | 1 | 9 | 36 | 84 | 126 | 126 | 84 | 36 | 9 | 1 | |
| 10 | 1 | 10 | 45 | 120 | 210 | 252 | 210 | 120 | 45 | 10 | 1 |

### 2.23.3 $S(n, m)$

| n \ m | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | | | | | | |
| 2 | 1 | 1 | | | | | | | | |
| 3 | 1 | 3 | 1 | | | | | | | |
| 4 | 1 | 7 | 6 | 1 | | | | | | |
| 5 | 1 | 15 | 25 | 10 | 1 | | | | | |
| 6 | 1 | 31 | 90 | 65 | 15 | 1 | | | | |
| 7 | 1 | 63 | 301 | 350 | 140 | 21 | 1 | | | |
| 8 | 1 | 127 | 966 | 1701 | 1050 | 266 | 28 | 1 | | |
| 9 | 1 | 255 | 3025 | 7770 | 6951 | 2646 | 462 | 36 | 1 | |
| 10 | 1 | 511 | 9330 | 34105 | 42525 | 22827 | 5880 | 750 | 45 | 1 |

### 2.23.4 $F(n, m)$

| n \ m | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | | | | | | |
| 2 | 1 | 2 | | | | | | | | |
| 3 | 1 | 2 | 3 | | | | | | | |
| 4 | 1 | 3 | 4 | 5 | | | | | | |
| 5 | 1 | 3 | 5 | 6 | 7 | | | | | |
| 6 | 1 | 4 | 7 | 9 | 10 | 11 | | | | |
| 7 | 1 | 4 | 8 | 11 | 13 | 14 | 15 | | | |
| 8 | 1 | 5 | 10 | 15 | 18 | 20 | 21 | 22 | | |
| 9 | 1 | 5 | 12 | 18 | 23 | 26 | 28 | 29 | 30 | |
| 10 | 1 | 6 | 14 | 23 | 30 | 35 | 38 | 40 | 41 | 42 |

### 2.23.5 Functions

```
// + mod if needed!
ll fun1(ll n, ll m) {
    ll ret = 1LL;
    while (m) {
        if (m & 1) ret *= n;
        n = n * n;
        m >>= 1;
    }
    return ret;
}
ll fun2(ll n, ll m) {
    ll ret = S[n][m];
    for (ll i = 2LL; i <= m; ++i) ret *= i;
    return ret;
}
ll fun3(ll n, ll m) {
    ll ret = 0;
    for (ll i = min(n, m); i >= 1; --i) {
        ret += S[n][i];
    }
    return ret;
}
ll fun4(ll n, ll m) { return S[n][m]; }
ll fun5(ll n, ll m) { return C[n + m - 1][n]; }
ll fun6(ll n, ll m) { return C[n - 1][m - 1]; }
ll fun7(ll n, ll m) { return F[n][m]; }
ll fun8(ll n, ll m) { return F[n - m][m]; }
```

## 2.24 $A^x\%C == B$ (by ac)

```
typedef long long LL;
const int maxn = 65535;
struct hash
{
    int a,b,next;
} Hash[maxn << 1];
int flg[maxn + 66];
int top,idx;
void ins(int a,int b)
{
    int k = b & maxn;
    if(flg[k] != idx)
    {
        flg[k] = idx;
        Hash[k].next = -1;
        Hash[k].a = a;
        Hash[k].b = b;
        return ;
    }
    while(Hash[k].next != -1)
    {
        if(Hash[k].b == b) return ;
        k = Hash[k].next;
    }
    Hash[k].next = ++ top;
    Hash[top].next = -1;
    Hash[top].a = a;
    Hash[top].b = b;
}
int find(int b)
{
    int k = b & maxn;
    if(flg[k] != idx) return -1;
    while(k != -1)
    {
        if(Hash[k].b == b) return Hash[k].a;
        k = Hash[k].next;
    }
    return -1;
```

```
}
int gcd(int a,int b)
{
    return b?gcd(b,a%b):a;
}
int ext_gcd(int a,int b,int& x,int& y)
{
    int t,ret;
    if (!b)
    {
        x=1,y=0;
        return a;
    }
    ret=ext_gcd(b,a%b,x,y);
    t=x,x=y,y=t-a/b*y;
    return ret;
}
int Inval(int a,int b,int n)
{
    int x,y,e;
    ext_gcd(a,n,x,y);
    e=(LL)x*b%n;
    return e<0?e+n:e;
}
int pow_mod(LL a,int b,int c)
{
    LL ret=1%c;
    a%=c;
    while(b)
    {
        if(b&1)ret=ret*a%c;
        a=a*a%c;
        b>>=1;
    }
    return ret;
}
int BabyStep(int A,int B,int C)
{
    top = maxn;
    ++ idx;
    LL buf=1%C,D=buf,K;
    int i,d=0,tmp;
    for(i=0; i<=100; buf=buf*A%C,++i)if(buf==B)return i;
    while((tmp=gcd(A,C))!=1)
    {
        if(B%tmp)return -1;
        ++d;
        C/=tmp;
        B/=tmp;
        D=D*A/tmp%C;
    }
    int M=(int)ceil(sqrt((double)C));
    for(buf=1%C,i=0; i<=M; buf=buf*A%C,++i)ins(i,buf);
    for(i=0,K=pow_mod((LL)A,M,C); i<=M; D=D*K%C,++i)
    {
        tmp=Inval((int)D,B,C);
        int w ;
        if(tmp>=0&&(w = find(tmp)) != -1)return i*M+w+d;
    }
    return -1;
}
int main()
{
    int A,B,C;
    while(scanf("%d%d%d",&A,&C,&B)!=EOF,A || B || C)
    {
        B %= C;
        int tmp=BabyStep(A,B,C);
        if(tmp<0)puts("No Solution");
        else printf("%d\n",tmp);
    }
    return 0;
}
```

## 2.25 HarmonicNumber (by rejudge)

```
// Concrete Mathematics
// http://mathworld.wolfram.com/HarmonicNumber.html
double H(int n)
{
    // Euler-Mascheroni Constant
    static const double gamma=0.57721566490153286060651209008240243104 2;
    // Hn = gamma + phi0(n + 1) // Digamma Function

    // Euler-Maclaurin Integration Formulas
    return gamma + log(n)
        + 1 / (2.0 * n)
        - 1 / (12.0 * n * n)
        + 1 / (120.0 * n * n * n * n)
        - 1 / (252.0 * n * n * n * n * n * n);
    // http://www.research.att.com/~njas/sequences/A006953
    // delta = epsilon / (240 * n ^ 8)
}
```

## 2.26 Gauss int (Enumrate the arguments)

```
#define maxn 22
using namespace std;
int mat[maxn][maxn];
int n, m;

bool fre[maxn]; int fs[maxn], fnt;
int x[maxn];
```

```
int cal(int r) {
    for (int i = r - 1, j = m - 1; i >= 0 && j >= 0; --i) {
        while (j >= 0 && fre[j]) --j;
        if (j >= 0) {
            x[j] = mat[i][m];
            for (int k = j + 1; k < m; ++k) {
                x[j] ^= (mat[i][k] && x[k]);
            }
            --j;
        }
    }
    int ret = 0;
    for (int i = 0; i < m; ++i) ret += x[i];
    return ret;
}
int solve(int r) {
    int mx = 1 << fnt;
    int ret = inf;
    for (int i = 0; i < mx; ++i) {
        if (__builtin_popcount(i) >= ret) continue;
        for (int j = 0; j < fnt; ++j) {
            if (i & (1 << j)) x[fs[j]] = 1;
            else x[fs[j]] = 0;
        }
        ret = min(ret, cal(r));
    }
    return ret;
}
int gauss() {
    memset(fre, false, sizeof fre); fnt = 0;
    int r, c, mr, mx;
    for (r = c = 0; r < n && c < m; ++r, ++c) {
        mx = 0, mr = -1;
        for (int i = r; i < n; ++i) {
            if (abs(mat[i][c]) > mx) {
                mx = abs(mat[i][c]);
                mr = i;
            }
        }
        if (!~mr) {
            fre[c] = true;
            fs[fnt++] = c;
            --r;
            continue;
        }
        else if (mr != r) {
            for (int j = c; j <= m; ++j) {
                swap(mat[r][j], mat[mr][j]);
            }
        }
        for (int i = r + 1; i < n; ++i) {
            if (!mat[i][c]) continue;
            for (int j = c; j <= m; ++j) {
                mat[i][j] ^= mat[r][j];
            }
        }
    }
    return solve(r);
}
```

## 2.27 Gauss (mod)

```
ll x[maxn];
void gauss() {
    int r, c, mr;
    ll mx;
    ll g, ta, tb;
    for (r = c = 0; r < n && c < m; ++r, ++c) {
        mr = -1, mx = 0;
        for (int i = r; i < n; ++i) {
            if ( _abs(mat[i][c]) > mx ) {
                mx = _abs(mat[i][c]);
                mr = i;
            }
        }
        if (!~mr) {
            --r;
            continue;
        }
        else if (mr != r) {
            for (int i = c; i <= m; ++i) {
                swap(mat[mr][i], mat[r][i]);
            }
        }
        for (int i = r + 1; i < n; ++i) {
            if (!mat[i][c]) continue;
            g = gcd(mat[r][c], mat[i][c]);
            ta = mat[r][c] / g;
            tb = mat[i][c] / g;
            for (int j = c; j <= m; ++j) {
                mat[i][j] = mat[r][j] * tb - mat[i][j] * ta;
                mat[i][j] %= mod;
            }
        }
    }
    //must have a solution
    ll t;
    for (int i = m - 1; i >= 0; --i) {
        t = mat[i][m];
        for (int j = i + 1; j < m; ++j) {
            t -= mat[i][j] * x[j];
            t %= mod;
        }
        x[i] = MLES(mat[i][i], t, mod);
```

```
     /*  for (ll j = 0; j < mod; ++j) {
           if ((mat[i][i] * j - t) % mod == 0) {
               x[i] = j;
               break;
           }
       }*/
    }
}
```

## 2.28 Gauss (double)

```
#define maxn 110
using namespace std;
double const eps = 1e-8;
int n, m;
double mat[maxn][maxn];
inline int sgn(double x) { return x < -eps ? -1 : x < eps ? 0 : 1; }
double x[maxn];

void gauss() {
    int r, c, mr;
    double mx, t;
    for (r = c = 0; r < n && c < m; ++r, ++c) {
        mr = -1, mx = eps;
        for (int i = r; i < n; ++i) {
            if (fabs(mat[i][c]) > mx) {
                mx = fabs(mat[i][c]);
                mr = i;
            }
        }
        if (!~mr) {
            --r;
            continue;
        }
        else {
            for (int i = c; i <= m; ++i) {
                swap(mat[r][i], mat[mr][i]);
            }
        }
        for (int i = r + 1; i < n; ++i) {
            if (sgn(mat[i][c]) == 0) continue;
            t = mat[i][c] / mat[r][c];
            for (int j = c; j <= m; ++j) {
                mat[i][j] -= mat[r][j] * t;
            }
        }
    }

    for (int i = r - 1; i >= 0; --i) {
        t = mat[i][m];
        for (int j = i + 1; j < m; ++j) {
            t -= x[j] * mat[i][j];
        }
        x[i] = t / mat[i][i];
    }
}
```

## 2.29 Gauss (Linear Base)

```
int gauss() {
     int i,j,k;
     j=0;
     for (i=m-1;i>=0;i--)
     {
          for (k=j;k<n;k++)
                  if ( (a[k]>>i)&1 )
                          break;
          if (k<n)
          {
                  swap(a[k],a[j]);
                  for (k=0;k<n;k++)
                          if (k!=j && ( (a[k]>>i)&1 ))
                                  a[k]^=a[j];
                  j++;
          }
     }
     return j;
}
//the Kth Xor
inline int fun(int k) {
    int res=0;
    for (int i=0;i<r;i++) {
            if ((k>>i)&1) {
                res ^= a[r-i-1];
            }
    }
    return res;
}

//exist x?
bool find(int x) {
    if (x == 0) return true;
    int now = 0;
    for (int i = 0; i < r; ++i) {
        now ^= a[i];
        if (now == x) return true;
        else if (now > x) {
            now ^= a[i];
        }
    }
    return false;
}
```

# 3 Graph

## 3.1 Basic

```
#define maxn 505
#define maxm 250050
using namespace std;
struct edges {
   int u, c, next;
} e[maxm];
int p[maxn], idx;
int n, m; // |V|, |E|
void addedge(int u, int v, int c) {
    e[idx].u = v;
    e[idx].c = c;
    e[idx].next = p[u];
    p[u] = idx++;
}
void init() {
    idx = 0;
    memset(p, 0xff, sizeof(p));
}
```

## 3.2 Floyd

```
int n;
int mp[maxn][maxn]; //mp[][] = inf; mp[i][i] = 0;

void floyd(){
    for(int k=0;k<n;k++){
        for(int i=0;i<n;i++){
            if(i == k) continue;
            for(int j=0;j<n;j++){
                if(mp[i][k] + mp[k][j] < mp[i][j]) {
                    mp[i][j] = mp[i][k] + mp[k][j];
                }
            }
        }
    }
}
```

## 3.3 SPFA

```
int dist[maxn];
bool used[maxn];
queue<int> q;

void spfa(int s){
    int t, u, w;
    while(!q.empty()) q.pop();
    memset(used, false, sizeof(used));
    for(int i=0;i<n;i++) dist[i] = inf;
    dist[s] = 0;
    q.push(s);
    while(!q.empty()){
        t = q.front();
        q.pop();
        used[t] = false;
        for(int i=p[t];i!=-1;i=e[i].next){
            u = e[i].u;
            w = e[i].c;
            if(dist[t] + w < dist[u]){
                dist[u] = dist[t] + w;
                if(!used[u]){
                    used[u] = true;
                    q.push(u);
                }
            }
        }
    }
}
```

## 3.4 Dijkstra

```
struct node{
    int u, c;
    node (int u, int c) : u(u), c(c) {}
    node () {}
    friend bool operator <(node a, node b){
        return a.c > b.c;
    }
}tmp;
int dist[maxn];
bool used[maxn];
priority_queue<node> q;

void dijkstra(int s, int d){
    int t, u, w;
    while(!q.empty()) q.pop();
    memset(used, false, sizeof(used));
    for(int i=0;i<n;i++) dist[i] = inf;
    tmp = node(s, 0);
    dist[s]=0;
    q.push(tmp);
    while(!q.empty()){
        tmp = q.top();
        q.pop();
        t = tmp.u;
        if(used[t]) continue;
        else used[t] = true;
        if(t == d) return;
        for(int i=p[t];i!=-1;i=e[i].next){
            u = e[i].u;
```

```
            w = e[i].c;
            if(used[u]) continue;
            if(dist[t] + w < dist[u]){
                dist[u] = dist[t] + w;
                q.push( node(u, dist[u]) );
            }
        }
    }
}
```

## 3.5 Prim

```
#define maxn 101
using namespace std;
int mp[maxn][maxn];
bool inTree[maxn];
int min_length[maxn];

int prim(int n){
    int sum = 0;
    memset(inTree,false,sizeof(inTree));
    for(int i=1;i<n;i++) min_length[i] = inf;
    min_length[0] = 0;
    for(int i=0;i<n;i++){
        int min_index = -1;
        for(int j=0;j<n;j++){
          if(!inTree[j] &&
             (min_index == -1 || min_length[j] < min_length[min_index]) ){
             min_index = j;
          }
        }
        inTree[min_index] = true;
        sum += min_length[min_index];
        for(int j=0;j<n;j++){
           if(!inTree[j] && mp[j][min_index] < min_length[j] ){
                min_length[j] = mp[j][min_index];
           }
        }
    }
    return sum;
}
```

## 3.6 Sap

```
struct edges{
    int u,c,next;
}e[maxm];
int p[maxn],idx;
int n, m;

void addedge(int u,int v,int c,int cc=0){
    e[idx].u=v; e[idx].c=c; e[idx].next=p[u]; p[u]=idx++;
    e[idx].u=u; e[idx].c=cc; e[idx].next=p[v]; p[v]=idx++;

}
void init(){ idx=0; memset(p,0xff,sizeof(p));}

int gap[maxn],dis[maxn],pre[maxn],cur[maxn];

int sap(int s,int t){
    memset(dis,0,sizeof(dis));
    memset(gap,0,sizeof(gap));
    for(int i=1;i<=n;i++)cur[i]=p[i];
    int u=pre[s]=s, max_flow=0,step=inf;
    gap[0]=n;
    while(dis[s]<n){
loop:   for(int &i=cur[u];i!=-1;i=e[i].next){
            int v=e[i].u;
            if(e[i].c>0 && dis[u]==dis[v]+1){
                step=min(step,e[i].c);
                pre[v]=u;
                u=v;
                if(v==t){
                    max_flow += step;
                    for(u=pre[u];v!=s;v=u,u=pre[u]){
                        e[cur[u]].c -= step;
                        e[cur[u]^1].c += step;
                    }
                    step=inf;
                }
                goto loop;
            }
        }
        int mindis=n;
        for(int i=p[u];i!=-1;i=e[i].next){
            int v=e[i].u;
            if(e[i].c>0 && mindis>dis[v]){
                cur[u]=i;
                mindis=dis[v];
            }
        }
        if( (--gap[dis[u]])==0) break;
        gap[ dis[u] = mindis+1] ++;
        u=pre[u];
    }
    return max_flow;
}
```

## 3.7 Hungary_Matrix

```
int mat[maxn][maxn];
int matx[maxn],maty[maxn];
bool fy[maxn];
int N,M;
```

```
int path(int u){
   int v;
   for(v=0;v<M;v++){
      if(mat[u][v] && !fy[v]){
         fy[v]=1;
         if(maty[v]<0 || path(maty[v])){
            matx[u]=v;
            maty[v]=u;
            return 1;
         }
      }
   }
   return 0;
}
int hungary(){
   int res=0;
   memset(matx,0xff,sizeof(matx));
   memset(maty,0xff,sizeof(maty));
   for(int i=0;i<N;i++){
       if(matx[i]<0){
           memset(fy,false,sizeof(fy));
           res+=path(i);
       }
   }
   return res;
}
```

## 3.8 Cut-Vertex

```
int dfn[maxn], low[maxn], cnt[maxn], cont;
void dfs(int u, int pre) {
    int v;
    dfn[u] = low[u] = ++cont;
    for (int i = p[u]; ~i; i = e[i].next) {
        v = e[i].u;
        if (!dfn[v]) {
            dfs(v, pre);
            low[u] = min(low[u], low[v]);
            if (low[v] >= dfn[u]) ++cnt[u];
        }
        else {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (u != pre) ++cnt[u];
}
void init() {
    cont = 0;
    memset(dfn, 0, sizeof dfn);
    memset(cnt, 0, sizeof cnt);
}
// for (int i = 1; i <= n; ++i) if (!dfn[i]) dfs(i, i);
```

# 4 Data Structure and Others

## 4.1 LIS

```
int lis(int p){
    int len=0,low,high,mid;
    //dp[0]=-inf;
    for(int i=0;i<p;i++){
        low=1,high=len;
        while(low<=high){
           mid=(low+high)/2;
           if(a[i]>dp[mid])low=mid+1;
           else high=mid-1;
        }
        dp[low]=a[i];
        if(low>len)len++;
    }
    return len;
}
```

## 4.2 RMQ

```
//RMQ(max)
int dpm[20][maxn];
void init(int N){
    for(int i=1;i<=N;i++){dpm[0][i]=a[i];}
    for(int j=1;(1<<j)<=N;j++){
        for(int i=1;i+(1<<j)-1<=N;i++){
            dpm[j][i]=max(dpm[j-1][i],dpm[j-1][i+(1<<(j-1))]);
        }
    }
}
int getm(int a,int b){
    int k=(int)(log((double)(b-a+1))/log(2.0));
    return max(dpm[k][a],dpm[k][b-(1<<k)+1]);
}
```

## 4.3 Reversed number

```
int a[maxn], c[maxn];
__int64 ret;
void MergeSort(int l, int r) {
    if (l < r) {
        int mid = (l + r) >> 1;
        MergeSort(l, mid);
        MergeSort(mid + 1, r);
        int i = l, j = mid + 1, k = l;
        for (; i <= mid && j <= r; ) {
            if (a[i] <= a[j]) {
                c[k++] = a[i++];
            }
            else {
                ret += j - k;
                c[k++] = a[j++];
            }
        }
        while (i <= mid) c[k++] = a[i++];
        while (j <= r) c[k++] = a[j++];
        for (i = l; i <= r; ++i) a[i] = c[i];
    }
}
```

## 4.4 max_sum(plusplus)

```
using namespace std;
int a[1000001],b[1000001],num[1000001];
int main(){
    int M,N;
    while(scanf("%d%d",&M,&N)!=EOF && M && N){
        num[0]=0;
        for(int i=1;i<=N;i++)scanf("%d",&num[i]);
        memset(a,0,(N+1)*sizeof(a[0]));
        memset(b,0,(N+1)*sizeof(b[0]));
        int max;
        for(int i=1;i<=M;i++){
           max=0x80000000;
           for(int j=i;j<=N;j++){
               if(a[j-1]<b[j-1])a[j]=b[j-1]+num[j];
               else a[j]=a[j-1]+num[j];
               b[j-1]=max;
               if(a[j]>max)max=a[j];
           }
           b[N]=max;
        }
        printf("%d\n",max);
    }
    return 0;
}
```

## 4.5 Trie(52)

```
#define maxn 151
#define WORD_LEN 32
#define MAX_WORD 52
using namespace std;
struct Trie_Node{
    int id;
    Trie_Node *next[MAX_WORD];
    void init(){
         id=-1;
         memset(next,NULL,sizeof(next));
```

```
    }
}trie[maxn*WORD_LEN],root;
int tidx,cnt;
int insert(char* s){
    int i,j;
    Trie_Node *p=&root;
    for(i=0;s[i];i++){
        if(s[i]<='Z')j=s[i]-'A';
        else j=s[i]-'a'+26;
        if(p->next[j]==NULL){
            trie[tidx].init();
            p->next[j]=&trie[tidx++];
        }
        p=p->next[j];
    }
    if(p->id==-1)p->id=cnt++;
    return p->id;
}
void init(){
    root.init();
    tidx=cnt=0;
}
```

## 4.6 BinaryIndexedTree

```
struct binaryIndexedTrees{
    int num[maxn];
    void init(){
        memset(num,0,sizeof(num));
    }
    int lowbit(int x){
        return x&(-x);
    }
    void update(int p,int c){
        while(p<maxn){
            num[p] += c;
            p += lowbit(p);
        }
    }
    int sum(int p){
        int t=0;
        while(p>0){
            t += num[p];
            p -= lowbit(p);
        }
        return t;
    }
    int find_kth(int k){ // if (k > limit), return maxn; if (k < 0) return 1
        int now=0;
        for(int i=20;i>=0;i--){
            now |= (1<<i);
            if(now>=maxn || num[now]>=k){
                now ^= (1<<i);
            }
            else k -= num[now];
        }
        return now + 1;
    }
    int getkth2(int k){ //kth_2
        int l=0,r=maxn,mid,f;
        while(l<r-1){
            mid=(l+r)>>1;
            f=sum(mid);
            if(f>=k) r=mid;
            else l=mid;
        }
        return r;
    }
}bit;
```

## 4.7 Union_Set

```
int parents[maxn];
int Find(int a){
    return parents[a] < 0 ? a : parents[a] = Find(parents[a]);
}
void Union(int a,int b){
    if(parents[a] < parents[b]){ parents[a] += parents[b], parents[b] = a;}
    else{ parents[b] += parents[a], parents[a] = b;}
}
void init(){
    memset(parents, 0xff, sizeof(parents));
}
```

## 4.8 Union_Set(Vector)

```
int parents[maxn], v[maxn];
int Find(int a){
    if(parents[a] < 0) return a;
    else{
    int t = parents[a];
        parents[a] = Find(parents[a]);
        v[a] = (v[a] + v[t]) % LEN;
        return parents[a];
    }
}
void Union(int a,int b,int c){
    if(parents[a] < parents[b]){
        parents[a] += parents[b];
        parents[b] = a;
        v[b] = (v[b] + c) % LEN;
    }
    else{
        parents[b] += parents[a];
```

```
        parents[a] = b;
        v[a] = (v[a] - c + LEN) % LEN;
    }
}
Union(ra, rb, (v[a] - v[b] + c + LEN) % LEN); //addedge(b, a, c)
```

## 4.9 suffix_array

```
#define MAXL 100100
#define MAXC 256
using namespace std;
int arr[3][MAXL], cnt[MAXL], mc[MAXC], h[MAXL], *sa, *ta, *r, *tr, sz;
void sa_init(char *str, int len){
    sa = arr[0], ta = arr[1], r = arr[2], sz = 0;
    for(int i=0;i<len;i++) ta[i] = str[i];
    sort(ta, ta + len);
    for(int i=1;i<=len;i++){
        if(ta[i] != ta[i-1] || i == len) cnt[ mc[ ta[i-1] ] = sz++ ] = i;
    }
    for(int i=len-1;i>=0;i--) sa[ --cnt[ r[i] = mc[ str[i] ]]] = i;
    for(int k=1;k<len && r[sa[len-1]]<len-1;k<<=1){
        for(int i=0;i<len;i++) cnt[r[sa[i]]] = i + 1;
        for(int i=len-1;i>=0;i--) {
            if(sa[i] >= k) ta[ --cnt[ r[sa[i] - k] ] ] = sa[i] - k;
        }
        for(int i=len-k;i<len;i++) ta[ --cnt[r[i]] ] = i;
        tr = sa, sa = ta, tr[sa[0]] = 0;
        for(int i=1;i<len;i++) {
            tr[sa[i]] = tr[sa[i-1]] +
                (r[sa[i]] != r[sa[i-1]] || sa[i-1]+k >= len
                    || r[sa[i]+k] != r[sa[i-1]+k]);
        }
        ta = r, r = tr;
    }
}
void h_init(char *str, int len){
    for(int i=0,d=0,j;i<len;i++){
        if(str[i] == '#' || r[i] == len-1) h[r[i]] = d = 0; //'#' = 35
        else{
            if(d) d--;
            j = sa[r[i] + 1];
            while(str[i+d] != '#' && str[j+d] != '#'
                  && str[i+d] == str[j+d])
                    d++;
            h[r[i]] = d;
        }
    }
}
char str[MAXL];
```

## 4.10 sa_methods

```
Distinct Substrings = len * (len - 1) / 2 - sigma(i = 0..len - 1)(h[i])
```

## 4.11 RMQ(pos)

```
int a[maxn];
int lg[maxn], dpmax[20][maxn], dpmin[20][maxn];
int maxpos[20][maxn], minpos[20][maxn];
void rmq_init(int n){
    int i, j, k;
    for(lg[0]=-1,i=1;i<=n;i++){
        lg[i] = ((i & (i - 1)) == 0)? lg[i - 1] + 1: lg[i - 1];
        dpmax[0][i] = dpmin[0][i] = a[i];
        maxpos[0][i] = minpos[0][i] = i;
    }
    for(k=1;k<=lg[n];k++){
        for(i=1;i+(1<<k)-1<=n;i++){
            j = i + (1 << (k - 1));
            if(dpmax[k - 1][i] > dpmax[k - 1][j]){
                dpmax[k][i] = dpmax[k - 1][i];
                maxpos[k][i] = maxpos[k - 1][i];
            }
            else{
                dpmax[k][i] = dpmax[k - 1][j];
                maxpos[k][i] = maxpos[k - 1][j];
            }
            if(dpmin[k - 1][i] < dpmin[k - 1][j]){
                dpmin[k][i] = dpmin[k - 1][i];
                minpos[k][i] = minpos[k - 1][i];
            }
            else{
                dpmin[k][i] = dpmin[k - 1][j];
                minpos[k][i] = minpos[k - 1][j];
            }
        }
    }
}
int getMax(int a, int b){
    int t = lg[b - a + 1], p = b - (1 << t) + 1;
    return max(dpmax[t][a], dpmax[t][p]);
}
int getMin(int a, int b){
    int t = lg[b - a + 1], p = b - (1 << t) + 1;
    return min(dpmin[t][a], dpmin[t][p]);
}
int getMaxpos(int a, int b){
    int t = lg[b - a + 1], p = b - (1 << t) + 1;
    if(dpmax[t][a] > dpmax[t][p]) return maxpos[t][a];
    else return maxpos[t][p];
}
int getMinpos(int a, int b){
    int t = lg[b - a + 1], p = b - (1 << t) + 1;
    if(dpmin[t][a] < dpmin[t][p]) return minpos[t][a];
```

```
    else return minpos[t][p];
}
```

## 4.12 lcp

```
int RMQ[MAXL];
int mm[MAXL];
int best[20][MAXL];
void initRMQ(int n)
{
     int i,j,a,b;
     for(int i=1;i<=n;i++)RMQ[i] = h[i-1];
     for(mm[0]=-1,i=1;i<=n;i++)
     mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
     for(i=1;i<=n;i++) best[0][i]=i;
     for(i=1;i<=mm[n];i++)
     for(j=1;j<=n+1-(1<<i);j++)
     {
       a=best[i-1][j];
       b=best[i-1][j+(1<<(i-1))];
       if(RMQ[a]<RMQ[b]) best[i][j]=a;
       else best[i][j]=b;
     }
     return;
}
int askRMQ(int a,int b){
    int t;
    t=mm[b-a+1];b-=(1<<t)-1;
    a=best[t][a];b=best[t][b];
    return RMQ[a]<RMQ[b]?a:b;
}
int lcp(int a,int b)
{
    //if(a == b) return len - a;
    int t;
    a=r[a];b=r[b];
    if(a>b) {t=a;a=b;b=t;}
    return(h[askRMQ(a+1,b) - 1]);
}
```

## 4.13 KMP

```
int const maxn = 100100;
char s[maxn], p[maxn];
int fail[maxn], len;
void buildF(char *p) {
    for (int i = 1, j = fail[0] = ~0; i < len; fail[i++] = j += p[j + 1] ==
         p[i])
        while (~j && p[j + 1] != p[i]) j = fail[j];
}
int kmp(char *s, char *p) {
    int ret = 0;
    for (int i = 0, j = -1; s[i]; ++i) {
        while (~j && p[j + 1] != s[i]) j = fail[j];
        if (p[j + 1] == s[i]) ++j;
        if (j == len - 1) {
            ++ret;
            j = fail[j];
        }
    }
    return ret;
}
```

## 4.14 extKMP

```
int ext[maxn]; // lcp(pat's suffix, pat)
int ex[maxn]; // lcp(pat's suffix, str)
//exp. str = "aaaba", pat = "aba", then ex[] = {1, 1, 3, 0, 1}, ext[] = {3,
     0, 1}
//la = strlen(str), lb = strlen(pat);
void extkmp(char *str, char *pat, int ext[], int ex[]) {
    int p=0,k=1;
    while(pat[p] == pat[p+1]) p++;
    ext[0] = lb, ext[1] = p;
    for(int i=2;i<lb;i++){
        int x = k + ext[k] - i, y = ext[i - k];
        if (y < x) ext[i] = y;
        else{
            p = max(0, x);
            while (pat[p] == pat[p+i]) p++;
            ext[i] = p;
            k = i;
        }
    }
    p = k = 0;
    while(str[p] && str[p] == pat[p]) p++;
    ex[0] = p;
    for(int i=1;i<la;i++){
        int x = k + ex[k] - i, y = ext[i - k];
        if (y < x) ex[i] = y;
        else{
            p = max(0, x);
            while (pat[p] && pat[p] == str[p+i]) p++;
            ex[i] = p;
            k = i;
        }
    }
}
```

## 4.15 Manacher

```
// "aaa" -> "!#a#a#a#"
int p[MAXL], len;
char str[MAXL];
```

```
int pk(){
    int id, mx = 0, res = 0;
    for(int i=0;i<len;i++){
        if(mx > i) p[i] = min(p[2*id-i], mx-i);
        else p[i] = 1;
        for(;str[i+p[i]]==str[i-p[i]];p[i]++);
        res = max(res, p[i]);
        if(p[i] + i > mx){
            mx = p[i] + i;
            id = i;
        }
    }
    return res - 1;
}
```

## 4.16 Lower Representation

```
char str[MAXL];
int fun(){
    int n = strlen(str);
    int i = 0, j = 1, len = 0, x, y;
    while(i < n && j < n && len < n){
        x = i + len; if(x >= n) x -= n;
        y = j + len; if(y >= n) y -= n;
        if(str[x] == str[y]) len++;
        else if(str[x] < str[y]){
            j += len + 1;
            len = 0;
        }
        else{
            i = j;
            j++;
            len = 0;
        }
    }
    return i;
}
```

## 4.17 lisan

```
int arr[maxn], rk[maxn], mp[maxn];
int n, mx;
bool cmp(int a, int b){
    return arr[a] < arr[b];
}
void lisan(){
    for(int i=1;i<=n;i++) rk[i] = i;
    sort(rk + 1, rk + n + 1, cmp);
    mp[1] = arr[rk[1]];
    arr[rk[1]] = mx = 1;
    for(int i=2;i<=n;i++){
        if(arr[rk[i]] == mp[mx]) arr[rk[i]] = mx;
        else mp[++mx] = arr[rk[i]], arr[rk[i]] = mx;
    }
}
```

## 4.18 Aho-corasick (trie graph)

```
int root, idx;
struct trie_node{
    int next[size];
    int fail;
    bool flag;
    void init(){
        fail = -1, flag = false;
        memset(next, 0, sizeof(next));
    }
}trie[maxn * leng];
int q[maxn * leng];
void trie_init(){
    root = idx = 0;
    trie[root].init();
}
void insert(char *s){
    int i, j, p = root;
    for(i=0;s[i];i++){
        j = s[i] - 'A';
        if(!trie[p].next[j]){
            trie[++idx].init();
            trie[p].next[j] = idx;
        }
        p = trie[p].next[j];
    }
    trie[p].flag = true;
}
void build(){
    int j, p;
    q[0] = root;
    for(int l=0,h=1;l<h;){
        p = q[l++];
        for(j=0;j<size;j++){
            if(trie[p].next[j]){
                q[h++] = trie[p].next[j];
                if(trie[p].fail == -1)
                    trie[trie[p].next[j]].fail = root;
                else{
                    trie[trie[p].next[j]].fail =
                        trie[trie[p].fail].next[j];

                    trie[trie[p].next[j]].flag |=
                        trie[trie[trie[p].fail].next[j]].flag;
                }
            }
            else{
```

```
            if(trie[p].fail != -1)
                trie[p].next[j] = trie[trie[p].fail].next[j];
        }
    }
}
```

## 4.19 Matrixs

```
typedef long long ll;
ll const P = 1000000007LL;
int const maxn = 105;
struct matrix{
    int N;
    ll mat[maxn][maxn];
    void init(){
        scanf("%d", &N);
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
                scanf("%I64d", &mat[i][j]);
            }
        }
    }
    matrix operator+(matrix B){
        matrix C;
        C.N=N;
        for(int i=0;i<N;i++){
            for(int j=0;j<B.N;j++){
                C.mat[i][j]=(mat[i][j]+B.mat[i][j])%P;
            }
        }
        return C;
    }
    matrix operator *(matrix B){
        matrix C;
        C.N=N;
        memset(C.mat,0,sizeof(C.mat));
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
                if(mat[i][j]){
                   for(int k=0;k<N;k++){
                       C.mat[i][k]=(C.mat[i][k]+mat[i][j]*B.mat[j][k])%P;
                   }
                }
            }
        }
        return C;
    }
    matrix operator ^(int n){
        matrix C;
        C.N=N;
        memset(C.mat,0,sizeof(C.mat));
        for(int i=0;i<N;i++)C.mat[i][i]=1;
        while(n){
            if(n&1)C=C*(*this);
            *this=(*this)*(*this);
            n>>=1;
        }
        return C;
    }
    void print(){
        for(int i=0;i<N;i++){
            for(int j=0;j<N;j++){
                if(j == N - 1) cout<<mat[i][j]<<endl;
                else cout<<mat[i][j]<<"␣";
            }
        }
    }
}A,B,C;
```

## 4.20 to sum_Matrix

```
matrix convert(matrix A){ //
    matrix C;
    C.N=A.N*2;
    memset(C.mat,0,sizeof(C.mat));
    for(int i=0;i<A.N;i++){
        for(int j=0;j<A.N;j++){
            C.mat[i][j]=A.mat[i][j];
        }
    }
    for(int i=0;i<A.N;i++){
        C.mat[i][A.N+i]=1;
        C.mat[A.N+i][A.N+i]=1;
    }
    return C;
}
```

## 4.21 Recycling_Matrix

```
struct matrix{
    int n;
    ll mat[maxn];
    void init(){
        for(int i=0;i<n;i++) scanf("%I64d", &mat[i]);
    }
    matrix operator*(matrix B){
        matrix C;
        C.n = n;
        for(int i=0;i<n;i++){
            C.mat[i] = 0;
            for(int j=0;j<n;j++){
```

```
            if(i - j >= 0) C.mat[i] += mat[j] * B.mat[i - j];
            else C.mat[i] += mat[j] * B.mat[i - j + n];
        }
        C.mat[i] %= mod;
    }
    return C;
}
matrix operator^(int m){
    matrix C;
    C.n = n;
    memset(C.mat, 0, sizeof(C.mat));
    C.mat[0] = 1;
    while(m){
        if(m & 1) C = C * (*this);
        *this = (*this) * (*this);
        m >>= 1;
    }
    return C;
}
void print(){
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cout<<mat[(i - j + n) % n]<<"␣";
        }
        cout<<endl;
    }
}
}A, B, C;
```

## 4.22 $solve\,(k,n) = 1^k + 2^k + ...n^k$

```
typedef long long ll;
ll const P = 1000000007LL;
int const maxn = 105;
struct matrix {
    //...
    void init(int k) {
        memset(mat, 0, sizeof mat);
        N = k + 2;
        for (int i = 0; i < k + 1; ++i) {
            mat[i][0] = 1;
            for (int j = 1; j <= i; ++j) {
                mat[i][j] = mat[i - 1][j - 1] + mat[i - 1][j];
            }
        }
        for (int j = 0; j < k + 1; ++j) {
            mat[k + 1][j] = mat[k][j];
        }
        mat[k + 1][k + 1] = 1;
    }
    //...
}A, B, C;

ll solve(int k, int n) {
    if (n == 0) return 0;
    A.init(k);
    B.N = k + 2;
    memset(B.mat, 0, sizeof B.mat);
    for (int i = 0; i < B.N; ++i) B.mat[i][0] = 1;
    A = (A ^ (n - 1)) * B;
    return A.mat[k + 1][0];
}
```

## 4.23 HashMap

```
int const maxh = 1000000;
struct HashMap{
    int p[maxh], v[maxh], next[maxh], idx;
    ll dp[maxh];
    void init(){
        idx = 0;
        memset(p, 0xff, sizeof p);
    }
    void add(int u, ll val){
        int x = u % maxh;
        for(int i=p[x];i!=-1;i=next[i]){
            if(v[i] == u){
                dp[i] += val;
                return;
            }
        }
        dp[idx] = val;
        v[idx] = u;
        next[idx] = p[x];
        p[x] = idx++;
    }
} hm[2], *src, *des;
```

## 4.24 SegTree (add, renew, max, min)

```
#define inf 0x3f3f3f3f
#define Inf 0x3FFFFFFFFFFFFFFFLL
#define maxn 100100
using namespace std;
typedef long long ll;

int n, m;
int arr[maxn];
struct node {
    ll a;
    ll mx, mi;
    ll s, s2;
    int delta;
    void init(int flag, int d, ll x) {
```

```
        if (flag == 1) {
            delta = 1;
            a = x;
            s = x * d;
            s2 = x * x * d;
            mx = mi = x;
        }
        else if (flag == 2) {
            delta = 2;
            a += x;
            s2 += 2LL * x * s + x * x * d;
            s += x * d;
            mx += x, mi += x;
        }
    }
} tree[maxn << 2];

inline void pushUp(int p, int lp, int rp) {
    tree[p].s = tree[lp].s + tree[rp].s;
    tree[p].s2 = tree[lp].s2 + tree[rp].s2;
    tree[p].mx = max(tree[lp].mx, tree[rp].mx);
    tree[p].mi = min(tree[lp].mi, tree[rp].mi);
}
inline void pushDown(int p, int lp, int rp, int l, int r, int mid) {
//    printf("pd(%d,%d,%d,%d,%d,%d)\n",p,lp,rp,l,r,mid);
    if (tree[p].delta != 0) {
        if (tree[p].delta == 1) {
            tree[lp].init(1, mid - l + 1, tree[p].a);
            tree[rp].init(1, r - mid, tree[p].a);
            tree[p].delta = 0;
            tree[p].a = 0;
        }
        else {
            if (tree[lp].delta == 1) {
                tree[lp].init(1, mid - l + 1, tree[lp].a + tree[p].a);
            }
            else tree[lp].init(2, mid - l + 1, tree[p].a);
            if (tree[rp].delta == 1) {
                tree[rp].init(1, r - mid, tree[rp].a + tree[p].a);
            }
            else tree[rp].init(2, r - mid, tree[p].a);
            tree[p].delta = 0;
            tree[p].a = 0;
        }
    }

}
void build(int l, int r, int p) {
    if (l == r) {
        tree[p].s = tree[p].mx = tree[p].mi = arr[l];
        tree[p].s2 = arr[l] * arr[l];
        tree[p].delta = 0;
        tree[p].a = 0;
        return;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
    build(l, mid, lp);
    build(mid + 1, r, rp);

    pushUp(p, lp, rp);
    tree[p].delta = 0;
    tree[p].a = 0;
}

void update_renew(int l, int r, int a, int b, ll c, int p) {
    if (l == a && r == b) {
        tree[p].init(1, r - l + 1, c);
        return;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
    pushDown(p, lp, rp, l, r, mid);
    if (b <= mid) update_renew(l, mid, a, b, c, lp);
    else if (a > mid) update_renew(mid + 1, r, a, b, c, rp);
    else {
        update_renew(l, mid, a, mid, c, lp);
        update_renew(mid + 1, r, mid + 1, b, c, rp);
    }
    pushUp(p, lp, rp);
}

void update_add(int l, int r, int a, int b, ll c, int p) {
    if (l == a && r == b) {
        if (tree[p].delta == 1) {
            tree[p].init(1, r - l + 1, c + tree[p].a);
        }
        else tree[p].init(2, r - l + 1, c);
        return;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
    pushDown(p, lp, rp, l, r, mid);
    if (b <= mid) update_add(l, mid, a, b, c, lp);
    else if (a > mid) update_add(mid + 1, r, a, b, c, rp);
    else {
        update_add(l, mid, a, mid, c, lp);
        update_add(mid + 1, r, mid + 1, b, c, rp);
    }
    pushUp(p, lp, rp);
}

ll query_s(int l, int r, int a, int b, int p) {
    if (l == a && r == b) {
        return tree[p].s;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
```

```
    pushDown(p, lp, rp, l, r, mid);
    if (b <= mid) return query_s(l, mid, a, b, lp);
    else if (a > mid) return query_s(mid + 1, r, a, b, rp);
    else return query_s(l, mid, a, mid, lp) + query_s(mid + 1, r, mid + 1,
        b, rp);
}
ll query_s2(int l, int r, int a, int b, int p) {
    if (l == a && r == b) {
        return tree[p].s2;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
    pushDown(p, lp, rp, l, r, mid);
    if (b <= mid) return query_s2(l, mid, a, b, lp);
    else if (a > mid) return query_s2(mid + 1, r, a, b, rp);
    else return query_s2(l, mid, a, mid, lp) + query_s2(mid + 1, r, mid + 1,
        b, rp);
}
ll query_mx(int l, int r, int a, int b, int p) {
    if (l == a && r == b) {
        return tree[p].mx;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
    pushDown(p, lp, rp, l, r, mid);
    if (b <= mid) return query_mx(l, mid, a, b, lp);
    else if (a > mid) return query_mx(mid + 1, r, a, b, rp);
    else return max( query_mx(l, mid, a, mid, lp), query_mx(mid + 1, r, mid
        + 1, b, rp) );
}
ll query_mi(int l, int r, int a, int b, int p) {
    if (l == a && r == b) {
        return tree[p].mi;
    }
    int mid = (l + r) >> 1, lp = p << 1, rp = p << 1 | 1;
    pushDown(p, lp, rp, l, r, mid);
    if (b <= mid) return query_mi(l, mid, a, b, lp);
    else if (a > mid) return query_mi(mid + 1, r, a, b, rp);
    else return min( query_mi(l, mid, a, mid, lp), query_mi(mid + 1, r, mid
        + 1, b, rp) );
}
```

## 4.25 Split tree

```
#define inf 0x3f3f3f3f
#define Inf 0x3FFFFFFFFFFFFFFFLL
#define maxn 100100
using namespace std;
int num[20][maxn];
int leftcnt[20][maxn];
int sd[maxn];
void build(int l, int r, int d){
    if(l == r) return;
    int mid = (l + r) >> 1;
    int lsame = mid - l + 1;
    for(int i=l;i<=r;i++)if(num[d][i] < sd[mid]) lsame--;
    int lp = l, rp = mid + 1;
    for(int i=l;i<=r;i++){
        if(i == l) leftcnt[d][i] = 0;
        else leftcnt[d][i] = leftcnt[d][i - 1];
        if(num[d][i] < sd[mid]){
            num[d + 1][lp++] = num[d][i];
            leftcnt[d][i]++;
        }
        else if(num[d][i] > sd[mid]){
            num[d + 1][rp++] = num[d][i];
        }
        else{
            if(lsame){
                lsame--;
                num[d + 1][lp++] = num[d][i];
                leftcnt[d][i]++;
            }
            else{
                num[d + 1][rp++] = num[d][i];
            }
        }
    }
    build(l, mid, d + 1);
    build(mid + 1, r, d + 1);
}
int query(int l, int r, int a, int b, int k, int d){
    if(l == r) return num[d][l];
    int mid = (l + r) >> 1;
    int ct = leftcnt[d][b], lct = 0;
    if(l < a){
        ct -= leftcnt[d][a - 1];
        lct = leftcnt[d][a - 1];
    }
    if(ct >= k){
        return query(l, mid, l + lct, l + lct + ct - 1, k, d + 1);
    }
    else{
        k -= ct;
        ct = b - a + 1 - ct;
        lct = a - l - lct;
        return query(mid + 1, r, mid + 1 + lct, mid + lct + ct, k, d + 1);
    }
}
int main(){
    int n, m;
    int a, b, k;
    while(~scanf("%d%d", &n, &m)){
        for(int i=1;i<=n;i++){
            scanf("%d", &num[0][i]);
        }
```

```
        memcpy(sd, num[0], sizeof(num[0]));
        sort(sd + 1, sd + n + 1);
        build(1, n, 0);
        while(m--){
            scanf("%d%d%d", &a, &b, &k);
            printf("%d\n", query(1, n, a, b, k, 0));
        }
    }
    return 0;
}
```

## 4.26 Splay

```
#define maxn 200200
using namespace std;
struct node{
    int key, minv, size, delta, rev;
    node *ch[2], *pre;
    void add(int v){
        if(size == 0) return;
        key += v;
        minv += v;
        delta += v;
    }
    void reverse(){
        if(size == 0) return;
        rev ^= 1;
        swap(ch[0], ch[1]);
    }
    void update(){
        size = ch[0]->size + ch[1]->size + 1;
        minv = min(key, min(ch[0]->minv, ch[1]->minv));
    }
    void pushdown(){
        if(delta){
            ch[0]->add(delta);
            ch[1]->add(delta);
            delta = 0;
        }
        if(rev){
            ch[0]->reverse();
            ch[1]->reverse();
            rev = 0;
        }
    }
};
int num[maxn];
#define keytree root->ch[1]->ch[0]
struct SplayTree{
    int cnt, top;
    node *st[maxn], data[maxn], *root, *null;
    node* newnode(int v){
        node *p;
        if(top) p = st[top--];
        else p = &data[cnt++];
        p->key = p->minv = v;
        p->delta = p->rev = 0;
        p->size = 1;
        p->pre = p->ch[0] = p->ch[1] = null;
        return p;
    }
    void init(){
        cnt = top = 0;
        null = newnode(inf);
        null->size = 0;
        root = newnode(inf);
        root->ch[1] = newnode(inf);
        root->ch[1]->pre = root;
        root->update();
    }
    node* build(int l, int r){
        if(l > r) return null;
        int mid = (l + r) >> 1;
        node *p = newnode(num[mid]);
        p->ch[0] = build(l, mid - 1);
        p->ch[1] = build(mid + 1, r);
        if(p->ch[0] != null) p->ch[0]->pre = p;
        if(p->ch[1] != null) p->ch[1]->pre = p;
        p->update();
        return p;
    }
    // c=0 zag, c=1 zig
    void rotate(node *x, int c){
        node *y = x->pre;
        y->pushdown();
        x->pushdown();
        y->ch[!c] = x->ch[c];
        if(x->ch[c] != null) x->ch[c]->pre = y;
        x->pre = y->pre;
        if(y->pre != null) y->pre->ch[y == y->pre->ch[1]] = x;
        x->ch[c] = y;
        y->pre = x;
        y->update();
        if(y == root) root = x;
    }
    void splay(node *x, node *f){
        x->pushdown();
        while(x->pre != f){
            if(x->pre->pre == f){
                rotate(x, x->pre->ch[0] == x);
                break;
            }
            node *y = x->pre;
            node *z = y->pre;
```

```
            int c = (y == z->ch[0]);
            if(y->ch[c] == x){
                rotate(x, !c), rotate(x, c);
            }
            else{
                rotate(y, c), rotate(x, c);
            }
        }
        x->update();
    }
    void select(int k, node *x){
        node *p = root;
        int tmp;
        while(1){
            p->pushdown();
            tmp = p->ch[0]->size;
            if(tmp == k) break;
            else if(tmp < k){
                k -= tmp + 1;
                p = p->ch[1];
            }
            else p = p->ch[0];
        }
        splay(p, x);
    }
/*------------------------------------------------
ADD x y D: Add D to each number in sub-sequence {Ax ... Ay}.
For example, performing "ADD 2 4 1" on {1, 2, 3, 4, 5} results in {1, 3, 4,
    5, 5}
REVERSE x y: reverse the sub-sequence {Ax ... Ay}.
For example, performing "REVERSE 2 4" on {1, 2, 3, 4, 5} results in {1, 4,
    3, 2, 5}
REVOLVE x y T: rotate sub-sequence {Ax ... Ay} T times.
For example, performing "REVOLVE 2 4 2" on {1, 2, 3, 4, 5} results in {1,
    3, 4, 2, 5}
INSERT x P: insert P after Ax.
For example, performing "INSERT 2 4" on {1, 2, 3, 4, 5} results in {1, 2,
    4, 3, 4, 5}
DELETE x: delete Ax.
For example, performing "DELETE 2" on {1, 2, 3, 4, 5} results in {1, 3, 4,
    5}
MIN x y: query the minimum number in subsequence{Ax .. Ay}.
For example, the correct answer to "MIN 2 4" on {1, 2, 3, 4, 5} is 2
------------------------------------------------*/
    void add(int a, int b, int c){
        select(a - 1, null);
        select(b + 1, root);
        keytree->add(c);
        splay(keytree, null);
    }
    void reverse(int a, int b){
        select(a - 1, null);
        select(b + 1, root);
        keytree->reverse();
        splay(keytree, null);
    }
    void revolve(int a, int c, int d){
        int len = c - a + 1;
        d %= len; if(d < 0) d += len;
        int b = c - d;
        if(d == 0) return;
        else if(d == 1){
            del(c);
            insert(a - 1, st[top]->key);
        }
        else{
            select(b + 1, null);
            select(c + 1, root);
            select(a - 1, root);
            select(c, root->ch[1]);
            node *p = root->ch[0]->ch[1];
            root->ch[0]->ch[1] = null;
            root->ch[0]->update();
            keytree->ch[1] = p;
            p->pre = keytree;
            splay(p, null);
        }
    }
    void insert(int a, int c){
        select(a, null);
        select(a + 1, root);
        keytree = newnode(c);
        keytree->pre = root->ch[1];
        root->ch[1]->update();
        splay(keytree, null);
    }
    void del(int a){
        select(a, null);
        node *tr = root;
        root = root->ch[1];
        root->pre = null;
        select(0, null);
        root->ch[0] = tr->ch[0];
        root->ch[0]->pre = root;
        root->update();
        st[++top] = tr;
    }
    int getmin(int a, int b){
        select(a - 1, null);
        select(b + 1, root);
        int res = keytree->minv;
        splay(keytree, null);
        return res;
    }
```

```
    void debug() {vis(root);}
    void vis(node* t) {
        if (t == null) return;
        t -> pushdown();
        vis(t->ch[0]);
        printf("node%2d:lson␣%2d,rson␣%2d,pre␣%2d,sz=%2d,key=%2d\n",
               t - data, t->ch[0] - data, t->ch[1] - data,
               t->pre - data, t->size, t->key);
        vis(t->ch[1]);
    }
}spt;
int main(){
    int n;
    char op[20]; int x,y,z;
    while(~scanf("%d", &n)){
        for(int i=1;i<=n;i++) scanf("%d", &num[i]);
        spt.init();
        if(n > 0){
            node *tr = spt.build(1, n);
            spt.keytree = tr;
            tr->pre = spt.root->ch[1];
            spt.splay(tr, spt.null);
        }
        //spt.debug();
        ...
    }
    return 0;
}
```

## 4.27 Rectangles' Union Area

```
#define maxn 1010
using namespace std;
typedef long long ll;
int n;
struct node {
    ll _x1, _x2, y1, y2;
    int x1, x2;
} rec[maxn];

ll xpos[maxn];

int find1(int l, int r, ll x){ // a[res] <= x
    int mid;
    while(l <= r){
        mid = (l + r) >> 1;
        if(xpos[mid] <= x) l = mid + 1;
        else r = mid - 1;
    }
    return r;
}

struct lines{
    int l, r, flag;
    ll h;
    friend bool operator<(lines a, lines b){
        if(a.h == b.h) return a.flag < b.flag;
        else return a.h < b.h;
    }
}line[maxn];

struct tree_node{
    int cnt;
    ll s;
}tree[maxn * 4];

void build(int l, int r, int p){
    if(l == r){
        tree[p].cnt = 0;
        tree[p].s = 0;
        return;
    }
    int mid = (l + r) >> 1;
    build(l, mid, 2*p);
    build(mid+1, r, 2*p+1);
    tree[p].cnt = 0;
    tree[p].s = 0;
}
void node_update(int l, int r, int p, int lp, int rp){
    if(tree[p].cnt >= 1) tree[p].s = xpos[r] - xpos[l - 1];
    else if(l == r) tree[p].s = 0;
    else tree[p].s = tree[lp].s + tree[rp].s;
}
void update(int l, int r, int a, int b, int c, int p){
    int mid = (l + r) >> 1, lp = 2*p, rp = 2*p+1;
    if(l == a && r == b){
        tree[p].cnt += c;
        node_update(l, r, p, lp, rp);
        return;
    }
    if(b <= mid) update(l, mid, a, b, c, lp);
    else if(a > mid) update(mid+1, r, a, b, c, rp);
    else{
        update(l, mid, a, mid, c, lp);
        update(mid+1, r, mid+1, b, c, rp);
    }
    node_update(l, r, p, lp, rp);
}

int main() {
    int _ca = 1;
    while (scanf("%d", &n) && n) {
        int xnt = 0;
        for (int i = 0; i < n; ++i) {
```

```
            scanf("␣%lld␣%lld␣%lld␣%lld", &rec[i]._x1, &rec[i].y1,
                 &rec[i]._x2, &rec[i].y2);
            xpos[xnt++] = rec[i]._x1, xpos[xnt++] = rec[i]._x2;
        }
        sort(xpos, xpos + xnt);
        int cnt = 1;
        for (int i = 1; i < xnt; ++i) {
            if (xpos[i] != xpos[i - 1]) {
                xpos[cnt++] = xpos[i];
            }
        }
        for (int i = 0; i < n; ++i) {
            rec[i].x1 = find1(0, cnt - 1, rec[i]._x1) + 1;
            rec[i].x2 = find1(0, cnt - 1, rec[i]._x2) + 1;
        }
        int x1, x2; ll y1, y2;
        int N = n << 1;
        for (int i = 0; i < N; i += 2) {
            x1 = rec[i >> 1].x1;
            x2 = rec[i >> 1].x2;
            y1 = rec[i >> 1].y1;
            y2 = rec[i >> 1].y2;
            line[i].l = x1, line[i].r = x2, line[i].h = y1, line[i].flag = 1;
            line[i+1].l = x1, line[i+1].r = x2, line[i+1].h = y2,
                 line[i+1].flag = -1;
        }
        sort(line, line + N);
        build(1, cnt, 1);
        int a, b, c;
        ll ret = 0;
        for (int i = 0;i < N - 1; ++i) {
            a = line[i].l;
            b = line[i].r - 1;
            c = line[i].flag;
            update(1, cnt, a, b, c, 1);
            ret += tree[1].s * (line[i + 1].h - line[i].h);
        }
        printf("Test␣case␣#%d\nTotal␣explored␣area:␣%lld\n\n", _ca++, ret);

    }
    return 0;
}
```

## 4.28 Binary_searches

```
int find1(int l, int r, int x) { // a[res] <= x
    int mid;
    while (l <= r) {
        mid = (l + r) >> 1;
        if (a[mid] <= x) l = mid + 1;
        else r = mid - 1;
    }
    return r;
}
int find2(int l, int r, int x) { // a[res] < x
    int mid;
    while (l <= r) {
        mid = (l + r) >> 1;
        if (a[mid] < x) l = mid + 1;
        else r = mid - 1;
    }
    return r;
}
int find3(int l, int r, int x) { // a[res] >= x
    int mid;
    while (l <= r) {
        mid = (l + r) >> 1;
        if (a[mid] >= x) r = mid - 1;
        else l = mid + 1;
    }
    return l;
}
int find4(int l, int r, int x) { // a[res] > x
    int mid;
    while (l <= r) {
        mid = (l + r) >> 1;
        if (a[mid] > x) r = mid - 1;
        else l = mid + 1;
    }
    return l;
}
```

## 4.29 Trichotomy

```
double const eps = 1e-8;
inline double Calc(double x) {
    //...
}
double Solve(double mi, double mx) {
    double Left, Right;
    double mid, midmid;
    double midr, midmidr;
    Left = mi; Right = mx;
    while (Left + eps < Right) {
        mid = (Left + Right) / 2;
        midmid = (mid + Right) / 2;
        mid_area = Calc(mid);
        midmid_area = Calc(midmid);
        if (mid_area >= midmid_area) Right = midmid;
        else Left = mid;
    }
    return midmid_area; // or sth.
}
```

# 5 JAVA

## 5.1 Date

```
SimpleDateFormat df=new SimpleDateFormat("yyyy-MM-dd␣EEEE",Locale.US);
while(cin.hasNext()){
    n=cin.nextInt();
    if(n==-1)break;
    GregorianCalendar wt=new GregorianCalendar(2000,Calendar.JANUARY,1);
    wt.add(GregorianCalendar.DATE, n);
    Date d=wt.getTime();
    System.out.println(df.format(d));
}
```

## 5.2 JAVA_IO

```
public static String readtxt() throws IOException{
    BufferedReader br=new BufferedReader(new FileReader("d:/sql.txt"));
    String str="";
    String r=br.readLine();
    while(r!=null){
        str+=r;
        r=br.readLine();
    }
    return str;
}
```

## 5.3 Chinese_Theory

```
static BigInteger[] m, r; //mod[], a[]
static BigInteger X,Y;
static BigInteger f2(BigInteger a, BigInteger b){
    if(b.compareTo(BigInteger.ZERO)==0){
        X = BigInteger.ONE;
        Y = BigInteger.ZERO;
        return a;
    }
    BigInteger d = f2(b, a.mod(b));
    BigInteger t = X;
    X = Y;
    Y = t.subtract(a.divide(b).multiply(Y));
    return d;
}
static BigInteger gcd(BigInteger a, BigInteger b){
    if(b.compareTo(BigInteger.ZERO) == 0) return a;
    else return gcd(b, a.mod(b));
}
static BigInteger f1(int len){
    int i; boolean flag = false;
    BigInteger m2,r2,d,c,t;
    BigInteger m1 = m[0], r1 = r[0];
    for(i=0;i<len-1;i++){
        m2 = m[i+1];
        r2 = r[i+1];
        d = f2(m1, m2);
        c = r2.subtract(r1);
        if(c.mod(d).compareTo(BigInteger.ZERO) != 0){
        flag = true;
        break;
        }
        X = X.multiply(c).divide(d);
        t = m2.divide(d);
        X = (X.mod(t).add(t)).mod(t);
        r1 = m1.multiply(X).add(r1);
        m1 = m1.multiply(m2).divide(d);
    }
    if(flag == true){
        return BigInteger.ZERO;
    }
    else{
        if(r1.compareTo(BigInteger.ZERO)==0 && len > 1){
        r1 = m[0];
        for(i=1;i<len;i++)r1 = gcd(m[i],r1);
        BigInteger ans = BigInteger.ONE;
        for(i=0;i<len;i++) ans = ans.multiply(m[i]);
        r1 = ans.divide(r1);
        }
        if(r1.compareTo(BigInteger.ZERO)==0 && len==1) r1 = m[0];
        return r1;
    }
}

static BigInteger lcm(BigInteger a, BigInteger b){
    return a.divide(gcd(a,b)).multiply(b);
}
static BigInteger rec(int len){
    BigInteger res = BigInteger.ONE;
    for(int i=0;i<len;i++){
        res = lcm(res, m[i]);
    }
    return res;
}
```

## 5.4 Matrix

```
class Matrix {
    int n;
    BigInteger mat[][];
    void init(int k) {
        n = k + 2;
        for (int i = 0; i < k + 1; ++i) {
            mat[i][0] = BigInteger.ONE;
            for (int j = 1; j <= i; ++j) {
```

```
            mat[i][j] = mat[i - 1][j - 1].add
                        (mat[i - 1][j]);
        }
    }
    for (int j = 0; j < k + 1; ++j) {
        mat[k + 1][j] = mat[k][j];
    }
    mat[k + 1][k + 1] = BigInteger.ONE;
}
public Matrix() {}
public Matrix(int n) {
    this.n = n;
    this.mat = new BigInteger[n][n];
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            this.mat[i][j] = BigInteger.ZERO;
        }
    }
}
Matrix mul(Matrix a) {
    Matrix C = new Matrix(n);
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (mat[i][j].compareTo(BigInteger.ZERO) != 0) {
                for (int k = 0; k < n; ++k) {
                    C.mat[i][k] = C.mat[i][k].add
                                (this.mat[i][j].multiply
                                (a.mat[j][k]));
                }
            }
        }
    }
    return C;
}
Matrix pow(BigInteger m) {
    Matrix C = new Matrix(n);
    BigInteger two = BigInteger.ONE.add( BigInteger.ONE );
    for (int i = 0; i < n; ++i) C.mat[i][i] = BigInteger.ONE;
    while (m.compareTo(BigInteger.ZERO) != 0) {
        if (m.mod(two).compareTo(BigInteger.ZERO) != 0) {
            C = C.mul(this);
        }
        Matrix T = mul(this);
        this.mat = T.mat;
        m = m.divide(two);
    }
    return C;
}
void print() {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            System.out.print(mat[i][j] + "␣");
        }
        System.out.println();
    }
  }
}
```

//BigInteger comparator

```
Arrays.sort(arr, BigIntegerComparator.ascendingSort);

class BigIntegerComparator implements Comparator {

       // to sort in ascending order
       public static final BigIntegerComparator ascendingSort = new
            BigIntegerComparator(true);

       // to sort in descending order
       public static final BigIntegerComparator descendingSort = new
            BigIntegerComparator(false);

       // flag to handle ascending/descending mode
       private boolean isAscending;

       public int compare(Object o1, Object o2) {
              int resultFlag = 0;

              if ( (o1 instanceof BigInteger) && (o2 instanceof
                   BigInteger)) {
                     resultFlag =
                          ((BigInteger)o1).compareTo((BigInteger)o2);
              }

              // if we want descending we use -1 multiplier
              return (isAscending?1:-1)*resultFlag;
       }
       private BigIntegerComparator(boolean isAscending) {
              this.isAscending = isAscending;
       }
 }
```

# 6 Geometry

## 6.1 Circle_Intersection

```
#define Pi 3.14159265358979323846
using namespace std;
struct Circle
{
double r,x,y;
}a,b;

double distanc(Circle n,Circle m)
{
double dis=sqrt((n.x-m.x)*(n.x-m.x)+(n.y-m.y)*(n.y-m.y));
return dis;
}
double Areaone(Circle &M)
{
return M.r*M.r*Pi;
}

double Area(Circle A,Circle B)
{
double area=0.0;
Circle M=(A.r>B.r)?A:B;
Circle N=(A.r>B.r)?B:A;
double dis=distanc(M,N);
if((dis<M.r+N.r)&&(dis>M.r-N.r))
{
  double cosM1 = (M.r*M.r+dis*dis-N.r*N.r)/(2.0*M.r*dis);
  double cosN1 = (N.r*N.r+dis*dis-M.r*M.r)/(2.0*N.r*dis);
  double M1 = acos(cosM1); //arc
  double N1 = acos(cosN1);
  double TM =0.5*M.r*M.r*sin(2.0*M1); //area of tri
  double TN =0.5*N.r*N.r*sin(2.0*N1);
  double FM =(M1/Pi)*Areaone(M); //area of Fan-shaped
  double FN =(N1/Pi)*Areaone(N);
  area=FM+FN-TM-TN;
}
else if(dis<=M.r-N.r){
  area=Areaone(N);
}
return area;
}
```

## 6.2 cal_centre

```
double cal_center_x(double x1,double y1,double x2,double y2,double
     x3,double y3)
{
    return((y1*(y2*y2+x2*x2-y3*y3-x3*x3) - y2*(y1*y1 - y3*y3 + x1*x1 - x3*x3)
     + y3*(y1*y1-y2*y2+x1*x1-x2*x2))
     /(2*(-x1*y2 + x1*y3 + x2*y1 - x2*y3 - x3*y1 + x3*y2)));
}
double cal_center_y(double x1,double y1,double x2,double y2,double
     x3,double y3)
{
    return((x1*(x2*x2+y2*y2-x3*x3-y3*y3) - x2*(x1*x1 - x3*x3 + y1*y1 - y3*y3)
    + x3*(x1*x1-x2*x2+y1*y1-y2*y2))
    /(2*(-y1*x2 + y1*x3 + y2*x1 - y2*x3 - y3*x1 + y3*x2)));
}
```

## 6.3 Line_Intersection

```
const double eps=1e-8;
struct CPoint{double x,y;
}points[4],l1[2],l2[2];
int dcmp(double x){
   if(x<-eps)return -1;else return (x>eps);
}
double cross(CPoint p0,CPoint p1,CPoint p2){
   return (p1.x-p0.x)*(p2.y-p0.y)-(p2.x-p0.x)*(p1.y-p0.y);
}
int LineIntersection(CPoint p1,CPoint p2,CPoint p3,CPoint p4,CPoint &cp){
    double u=cross(p1,p2,p3),v=cross(p2,p1,p4);
    if(dcmp(u+v)){
        cp.x=(p3.x*v+p4.x*u)/(v+u);
        cp.y=(p3.y*v+p4.y*u)/(v+u);
        return 1;
    }
    if(dcmp(u))return 2; //none
    if(dcmp(cross(p3,p4,p1)))return 3;
    return -1; //line
}
```

## 6.4 Area of a Tetrahedron

```
//AB, AC, AD, CD, BD, BC.
double calc(double a, double b, double c, double r, double p, double q)
{
    a *= a, b *= b, c *= c, r *= r, p *= p, q *= q;
    double P1 = a * p * (-a + b + c - p + q + r);
    double P2 = b * q * (a - b + c + p - q + r);
    double P3 = c * r * (a + b - c + p + q - r);
    double P = a * b * r + a * c * q + b * c * p + p * q * r;
    return sqrt((P1 + P2 + P3 - P)) / 12.;
}
```

## 6.5 crosspoint(g++ better)

```
#include <complex>
```

```
#define eps (1e-8)
#define x real()
#define y imag()

using namespace std;
typedef complex<double> Point;
inline int sgn(double a){ return (a > eps) - (a < -eps);}
double cross(Point a, Point b){ return imag(conj(a) * b);}
double dmul(Point a, Point b){ return real(conj(a) * b);}
bool crosspoint(Point p1, Point p2, Point q1, Point q2){
    double a = cross(p2 - p1, q2 - q1), b = cross(p2 - p1, p2 - q1);
    double c = cross(q2 - q1, p2 - p1), d = cross(q2 - q1, q2 - p1);
    if(a == 0){
        return b != 0? 0:
            (sgn(dmul(q1 - p1, q1 - p2)) <= 0 ||
               sgn(dmul(q2 - p1, q2 - p2)) <= 0);}
    else
        return (sgn(b/a) >= 0 &&
            sgn(b/a - 1) <= 0 &&
            sgn(d/c) >= 0 &&
            sgn(d/c - 1) <= 0);

  // else return (sgn(d/c) >= 0 && sgn(d/c - 1) <= 0); cross on P
}
```

## 6.6 N Circles cover [1-K] times

```
#define maxn 105
using namespace std;

double const eps = 1e-8;
double const pi = atan2(0, -1.0);
inline int sgn(double x) { return x < -eps ? -1 : x < eps ? 0 : 1; }
struct pt {
    double x, y;
    pt (double _x, double _y) { x = _x, y = _y; }
    pt () {}
    pt operator+ (const pt a) { return pt(x + a.x, y + a.y); }
    pt operator- (const pt a) { return pt(x - a.x, y - a.y); }
    pt operator* (const double r) { return pt(x * r, y * r); }
    pt operator/ (const double r) { return pt(x / r, y / r); }
    inline void print() { printf("%.2lf␣%.2lf\n", x, y); }
} p[maxn];
inline double xmul(const pt &a, const pt &b) {
    return a.x * b.y - a.y * b.x;
}
inline double dist(const pt &a, const pt &b) {
    return sqrt( (a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y) );
}


int n;
double r[maxn];


inline int rlt(int a, int b) {
    double d = dist(p[a], p[b]), d1 = sgn(d - r[a] + r[b]),
           d2 = sgn(d - r[b] + r[a]);
    if (d1 < 0 || !d1 && (d > eps || a > b))return 0;
    if (d2 < 0 || !d2 && (d > eps || a < b))return 1;
    return d < r[a] + r[b] - eps ? 2 : 3;
}

double areaArc(pt &o, double r, double ang1, double ang2) {
    pt a(o.x + r * cos(ang1), o.y + r * sin(ang1));
    pt b(o.x + r * cos(ang2), o.y + r * sin(ang2));
    double dif = ang2 - ang1;
    return (xmul(a, b) + (dif - sin(dif)) * r * r) * 0.5;
}

pair<double, int> e[maxn << 1];
double res[maxn];
int cnt;

void cal() {
    fill(res, res + n + 1, 0.0);
    double last;
    pt X, Y;
    for (int i = 0; i < n; ++i) if (r[i] > eps) {
        int acc = 0;
        cnt = 0;
        e[cnt++] = make_pair(-pi, 1);
        e[cnt++] = make_pair(pi, -1);
        for (int j = 0; j < n; ++j) if (i != j && r[j] > eps) {
            int rel = rlt(i, j);
            if (rel == 1) {
                e[cnt++] = make_pair(-pi, 1);
                e[cnt++] = make_pair(pi, -1);
            }
            else if (rel == 2) {
                double center = atan2(p[j].y - p[i].y, p[j].x - p[i].x);
        double d2 = (p[i].x - p[j].x) * (p[i].x - p[j].x) +
                    (p[i].y - p[j].y) * (p[i].y - p[j].y);
        double ang = acos((r[i] * r[i] + d2 - r[j] * r[j]) /
                    (2 * r[i] * sqrt(d2)));
        double angX = center + ang;
        double angY = center - ang;
        if (angX > pi)angX -= 2 * pi;
        if (angY < -pi)angY += 2 * pi;

               if (angX < angY) ++acc;
               e[cnt++] = make_pair(angX, -1);
               e[cnt++] = make_pair(angY, 1);
        }
```

```
        }
        sort(e, e + cnt);
        last = -pi;
        for (int j = 0; j < cnt; ++j) {
            double tmp = areaArc(p[i], r[i], last, e[j].first);
            res[acc] += tmp;
            res[acc - 1] -= tmp;
            acc += e[j].second;
            last = e[j].first;
        }
    }
}
int main() {

    while (~scanf("%d", &n)) {
        for (int i = 0; i < n; ++i) {
            scanf("%lf␣%lf␣%lf", &p[i].x, &p[i].y, &r[i]);
        }
        cal();

    }
    return 0;
}
```

## 6.7 Graham(int)

```
typedef __int64 ll;
struct Point {
    ll x, y;
    friend bool operator < (Point a, Point b) {
        if (a.y == b.y) return a.x < b.x;
        else return a.y < b.y;
    }
} p[maxn], res[maxn];

ll Xmul(Point a, Point b, Point c) {
    return (b.x - a.x) * (c.y - a.y) - (c.x - a.x) * (b.y - a.y);
}

ll Xmul(Point b, Point c) {
    return b.x * c.y - c.x * b.y;
}
int Graham(Point pnt[], int n, Point res[]) {
    int i, j, top = 1;
    sort(pnt, pnt + n);
    pnt[n] = pnt[0];
    if (n == 0) return 0; res[0] = pnt[0];
    if (n == 1) return 1; res[1] = pnt[1];
    if (n == 2) return 2; res[2] = pnt[2];
    for (i = 2; i < n; ++i) {
        while (top && Xmul(res[top - 1], res[top], pnt[i]) <= 0) --top;
        res[++top] = pnt[i];
    }
    j = top;
    res[++top] = pnt[n - 2];
    for (i = n - 3; i >= 0; --i) {
        while (top != j && Xmul(res[top - 1], res[top], pnt[i]) <= 0) --top;
        res[++top] = pnt[i];
    }
    res[top] = res[0];
    return top;
}
```

## 6.8 Polar_Sort(convex)

```
#define maxn 1005

using namespace std;
struct Point{
    int x, y;
}p[maxn];

inline int cross(Point a, Point b){
    return a.x * b.y - a.y * b.x;
}
bool cmp(Point a, Point b){
    int t = cross(a, b);
    if(t == 0){
        if(a.x * b.x < 0 || a.y * b.y < 0){
            return a.y < b.y || a.y == b.y && a.x < b.x;
        }
        else{
            return abs(a.x) < abs(b.x) || abs(a.y) < abs(b.y);
        }
    }
    else return t > 0;
}
void polar_sort(int n){
    int mx = 0, x0, y0;
    for(int i=0;i<n;i++){
        if(p[i].x < p[mx].x) mx = i;
    }
    swap(p[0], p[mx]);
    x0 = p[0].x, y0 = p[0].y;
    for(int i=0;i<n;i++){
        p[i].x -= x0;
        p[i].y -= y0;
    }
    sort(p + 1, p + n, cmp);
    for(int i=n-1;i>=0;i--){
        if(cross(p[i], p[i-1]) != 0){
            reverse(p + i, p + n);
            break;
        }
```

```
    }
    for(int i=0;i<n;i++){
        p[i].x += x0;
        p[i].y += y0;
    }
}

int main(){
    int n;
    while(~scanf("%d", &n)){
        int mx = 0, x0, y0;
        for(int i=0;i<n;i++){
            scanf("%d%d", &p[i].x, &p[i].y);
        }
        polar_sort(n);
        for(int i=0;i<n;i++){
            printf("%d␣%d\n", p[i].x, p[i].y);
        }
    }
    return 0;
}
```

## 6.9 Ellipse's Circumference

```
double const pi = atan2(0, -1.0);

double cal(double a, double b) {
    double e2 = 1.0 - b * b / a / a;
    double e = e2;
    double ret = 1.0;
    double xa = 1.0, ya = 2.0;
    double t = 0.25;

    for (int i = 1; i <= 10000; ++i) {
        ret -= t * e;
        t = t * xa * (xa + 2) / (ya + 2) / (ya + 2);
        xa += 2.0;
        ya += 2.0;
        e *= e2;
    }
    return 2.0 * pi * a * ret;
}
int main() {
    int _ca = 1;
    double a, b;
    int T;
    for (scanf("%d", &T); T--; ) {
        scanf("%lf␣%lf", &a, &b);
        if (a < b) swap(a, b);
        printf("Case␣%d:␣%.10lf\n", _ca++, cal(a, b));
    }
    return 0;
}
```

## 6.10 Area of intersection between Convex & Circle

```
\\Centre of the circle (0, 0)

#define maxn 110

using namespace std;

#define sq(x) ((x) * (x))
#define sng(x) (x == 0.0? 0.0: (x > 0? 1.0: -1.0))
#define fmax(x, y) (x > y? x: y)
#define fmin(x, y) (x < y? x: y)

struct pt {
    double x, y;
    pt(double a = 0, double b = 0)
    {
        x = a;
        y = b;
    }
    double len() { return sqrt(sq(x) + sq(y)); }
    double operator * (pt o) { return x * o.y - o.x * y; }
    double operator % (pt o) { return x * o.x + y * o.y; }
} ps[maxn];

struct sg {
    pt a, b;
    double A, B, C;
    sg(pt x, pt y)
    {
        a = x;
        b = y;
        A = a.y - b.y;
        B = b.x - a.x;
        C = -(a.y * B + a.x * A);
    }
    bool ons(pt o){
        if (fmin(a.x, b.x) <= o.x && o.x <= fmax(a.x, b.x))
            if (fmin(a.y, b.y) <= o.y && o.y <= fmax(a.y, b.y))
                return 1;
        return 0;
    }
    double len() { return sqrt(sq(a.x - b.x) + sq(a.y - b.y)); }
    double ang() { return acos((a % b) / (a.len() * b.len())); }
    pt inr(sg o) {
        double d = (A * o.B - o.A * B);
        double x = B * o.C - o.B * C;
        double y = A * o.C - o.A * C;
        return pt(x / d, -y / d);
    }
```

```
};

double r;
int   n;

double TGL(pt a, pt b) { //Triangulate
    double sn = sng(a * b);
    if (a.len() < b.len())
        swap(a ,b);
    pt    lp(a.x - b.x, a.y - b.y), np(-lp.y, lp.x), cp;
    sg    l(a, b), nl(pt(0, 0), np);
    pt    tp = l.inr(nl);
    double tsu = 0;
    double oa = a.len();
    double ob = b.len();
    double ol = tp.len();;
    double ang, d;

    if (oa == 0.0 || oa == 0.0 || ol == 0.0)
        return 0.0;
    if (oa <= r && ob <= r)
    {
        tsu += fabs(a * b / 2.0);
    }
    else if (oa > r && ob <= r)
    {
        d = sqrt(sq(r) - sq(tp.len())) / l.len();
        tp = pt(tp.x + lp.x * d, tp.y + lp.y * d);
        ang = sg(a, tp).ang();
        tsu += ang * sq(r) / 2.0;
        tsu += fabs(tp * b/ 2.0);
    }
    else
    {
        ang = acos(ol / r);
        tsu += l.ang() * sq(r) / 2.0;
        if (oa > r && ob > r && ol < r && l.ons(tp))
            tsu += ol * r * sin(ang) - ang * sq(r);
    }

    return tsu * sn;
}

int main() {
    int i;
    double tsu;

    while (scanf("%lf", &r) != EOF)
    {
        scanf("%d", &n);
        for (i = 0; i < n; i++)
            scanf("%lf%lf", &ps[i].x, &ps[i].y);
        tsu = 0.0;
        for (i = 0; i < n; i++)
            tsu += TGL(ps[i], ps[(i + 1) % n]);
        printf("%.2lf\n", fabs(tsu));
    }

    return 0;
}
```

# 7 Others

```
//BigNum
/*
  Duze liczby z ustalana podstawa
*/

typedef unsigned long long digit;

#define MAX_DIGIT 1000000000
#define MAX_LENGTH 9 // MAX_DIGIT=10^MAX_LENGTH

class BigNum {
  vector<digit> data;

  void shrink() {
    while (data.size()>1 && !data[data.size()-1])
      data.resize(data.size()-1);
  }

  public:
    BigNum(digit i=0) {
      data.resize(1,i%MAX_DIGIT);
      i/=MAX_DIGIT;
      while (i) {
        data.resize(data.size()+1);
        data.back()=i%MAX_DIGIT;
        i/=MAX_DIGIT;
      }
    }

    explicit BigNum(const char *t) {
      int n=0,i,j,k;
      while (t[n])
        n++;
      for (i=n-1; i>=0; i-=MAX_LENGTH) {
        k=0;
        for (j=MAX_LENGTH-1; j>=0; j--)
          if (i-j>=0)
            k=10*k+t[i-j]-'0';
        data.push_back(k);
      }
      shrink();
    }

    BigNum &operator--() {
      int i=0;
      while (!data[i]) {
        data[i]=MAX_DIGIT-1;
        i++;
      }
      data[i]--;
      return *this;
    }

    BigNum &operator++() {
      int i=0;
      while (data[i]+1==MAX_DIGIT) {
        data[i]=0;
        i++;
      }
      data[i]++;
      return *this;
    }

    BigNum &operator+=(const BigNum &a) {
      digit i=0,p=0;
      while (p || i<data.size() || i<a.data.size()) {
        if (i<data.size())
          p+=data[i];
        if (i<a.data.size())
          p+=a.data[i];
        if (i>=data.size())
          data.resize(i+1);
        if (p>=MAX_DIGIT) {
          data[i]=p-MAX_DIGIT;
          p=1;
        }
        else {
          data[i]=p;
          p=0;
        }
        i++;
      }
      return *this;
    }

    BigNum &operator-=(const BigNum &a) {
      digit p=0;
      for (int i=0; i<data.size() || p; i++) {
        if (i<a.data.size())
          p+=a.data[i];
        if (p<=data[i]) {
          data[i]-=p;
          p=0;
        }
        else {
          data[i]+=MAX_DIGIT-p;
          p=1;
        }
      }
      shrink();
      return *this;
    }
```

```
BigNum operator+(BigNum a) {
  BigNum r=*this;

  r+=a;
  return r;
}

BigNum operator-(BigNum a) {
  BigNum r=*this;

  r-=a;
  return r;
}

digit operator%(digit d) {
  digit p=0;
  for (int i=data.size()-1; i>=0; i--)
    p=(p*MAX_DIGIT+data[i])%d;
  return p;
}

BigNum operator*(const BigNum &a) {
  BigNum r;
  if(zero()||a.zero())return r;
  for (int i=0; i<data.size(); i++) {
    BigNum t=a;
    t*=data[i];
    t.data.resize(t.data.size()+i);
    for (int j=t.data.size()-i-1; j>=0; j--)
      t.data[j+i]=t.data[j];
    for (int j=0; j<i; j++)
      t.data[j]=0;
    r+=t;
  }
  r.shrink();
  return r;
}

BigNum operator/(BigNum a) {
  BigNum ans,t=*this,power=1,ta=a;

  while (ta<t) {
    power*=10;
    ta*=10;
  }
  while (!power.zero()) {
    while (ta<t || ta==t) {
      ans+=power;
      t-=ta;
    }
    power/=10;
    ta/=10;
  }
  return ans;
}

BigNum operator%(BigNum a) {
  return *this-(*this/a)*a;
}

BigNum &operator*=(digit m) {
  digit p=0;
  for (int i=0; p || i<data.size(); i++) {
    if (i<data.size())
      p+=m*data[i];
    if (i>=data.size())
      data.resize(i+1);
    data[i]=p%MAX_DIGIT;
    p/=MAX_DIGIT;
  }
  return *this;
}

BigNum &operator/=(digit d) {
  digit p=0;
  for (int i=data.size()-1; i>=0; i--) {
    p=p*MAX_DIGIT+data[i];
    data[i]=p/d;
    p%=d;
  }
  shrink();
  return *this;
}

bool operator==(const BigNum &x) const {
  if (data.size()!=x.data.size())
    return false;
  int i=0;
  while (i<data.size() && data[i]==x.data[i])
    i++;
  return i==data.size();
}

bool operator<(const BigNum &x) const {
  if (x.data.size()!=data.size())
    return data.size()<x.data.size();
  int i=data.size()-1;
  while (i>=0 && data[i]==x.data[i])
    i--;
  return i>=0 && data[i]<x.data[i];
}

bool zero() const {
```

```
    return data.size()==1 && !data[0];
  }

  friend ostream &operator<<(ostream &out,const BigNum &a) {
    out<<a.data[a.data.size()-1];
    for (int i=a.data.size()-2; i>=0; i--) {
      digit j=a.data[i]+!a.data[i];
      while (j<MAX_DIGIT/10) {
        out<<0;
        j=j*10;
      }
     out<<a.data[i];
    }
    return out;
  }
};

struct euclid_result {
  BigNum alfa,beta,d;
  bool beta_negative;
  euclid_result(BigNum _alfa,BigNum _beta,BigNum _d,bool _beta_negative) {
    alfa=_alfa; beta=_beta; d=_d; beta_negative=_beta_negative;
  }
};

euclid_result extended_euclid(BigNum a,BigNum b) {
  if (b.zero())
    return euclid_result(1,0,a,true);
  euclid_result r=extended_euclid(b,a%b);
  // d=alfa*b+a%b*beta=a*beta+(-a/b+alfa)*b
  return euclid_result(r.beta,r.alfa+(a/b)*r.beta,r.d,!r.beta_negative);
}

BigNum inverse(BigNum a,BigNum m) {
  euclid_result r=extended_euclid(a,m);
  if (r.beta_negative)
    return r.alfa%m;
  else {
    return (m-r.alfa%m)%m;
  }
}
int main(){
   return 0;
}
```

## 7.1 BigNum

```
//bignum_uestc
const int maxleng=50;

class BigInt
{
public:
	int leng;
	int num[maxleng];
public:
	BigInt()
	{
		leng=1;
		memset(num,0,sizeof(num));
	}
	BigInt(int x)
	{
		leng=0;
		memset(num,0,sizeof(num));
		while(x)
		{
			num[leng++]=x%10000;
			x/=10000;
		}
		if(leng==0)leng=1;
	}
	operator int()
	{
		int x=0,l=leng-1;
		while(l>=0)
		{
			x=x*10000+num[l];
			l--;
		}
		return x;
	}
	operator int*()
	{
		return num;
	}
	int length()
	{
		return leng;
	}
	void read()
	{
		char s[maxleng+1];
		scanf("%s",s);
		int l=strlen(s);
		leng=0;
		for(int i=l-1;i>=0;)
		{
			if(i>=0)num[leng]+=(s[i--]-'0');
			if(i>=0)num[leng]+=(s[i--]-'0')*10;
			if(i>=0)num[leng]+=(s[i--]-'0')*100;
			if(i>=0)num[leng]+=(s[i--]-'0')*1000;
			leng++;
		}
```

```
		if(leng==0)leng=1;
	}
	void write()
	{
		int i=leng-1;
		printf("%d",num[i]);i--;
		while(i>=0)printf("%04d",num[i--]);
	}
	void writeln()
	{
		write();
		printf("\n");
	}
	void getlength()
	{
		leng=maxleng-1;
		while(num[leng]==0&&leng>0)leng--;
		leng++;
	}
	friend BigInt operator+(BigInt a,BigInt b);
	friend BigInt operator+(BigInt a,int b);
	friend BigInt operator-(BigInt a,BigInt b);
	friend BigInt operator*(BigInt a,BigInt b);
	friend BigInt operator*(BigInt a,int b);
	friend BigInt operator/(BigInt a,BigInt b);
	friend bool operator<(BigInt a,BigInt b);
};

BigInt operator+(BigInt a,BigInt b)
{
	int l=a.leng>b.leng?a.leng:b.leng,t=0;
	BigInt ans;
	for(int i=0;i<l;i++)
	{
		ans[i]=(a[i]+b[i]+t)%10000;
		t=(a[i]+b[i]+t)/10000;
	}
	while(t)
	{
		ans[l++]=t%10000;
		t/=10000;
	}
	ans.leng=l;
	return ans;
}

BigInt operator+(BigInt a,int b)
{
	int t=0;
	BigInt ans;
	memcpy(ans.num,a.num,sizeof(a.num));
	ans[t]+=b;
	while(a[t]>=10000)
	{
		ans[t+1]+=ans[t]/10000;
		ans[t]%=10000;
	}
	ans.getlength();
	return ans;
}

//a >= b
BigInt operator-(BigInt a,BigInt b)
{
	int l=a.leng;
	BigInt ans;
	memcpy(ans.num,a.num,sizeof(a.num));
	for(int i=0;i<l;i++)
	{
		ans[i]-=b[i];
		if(ans[i]<0)
		{
			ans[i]+=10000;
			ans[i+1]--;
		}
	}
	ans.getlength();
	return ans;
}

BigInt operator*(BigInt a,BigInt b)
{
	int la=a.leng,lb=b.leng,t,p;
	BigInt ans;
	for(int i=0;i<la;i++)
	{
		t=0;
		for(int j=0;j<lb;j++)
		{
			p=(ans[i+j]+a[i]*b[j]+t)/10000;
			ans[i+j]=(ans[i+j]+a[i]*b[j]+t)%10000;
			t=p;
		}
		p=i+lb;
		if(t)
		{
			ans[p]+=t;
			while(ans[p]>=10000)
			{
				ans[p+1]+=ans[p]/10000;
				ans[p]%=10000;
				p++;
			}
		}
```

```
        }
        ans.getlength();
        return ans;
}

BigInt operator*(BigInt a,int b)
{
        int t=0,p=a.leng;
        BigInt ans;
        for(int i=0;i<p;i++)
        {
                ans[i]=(a[i]*b+t)%10000;
                t=(a[i]*b+t)/10000;
        }
        while(t)
        {
                ans[p++]=t%10000;
                t/=10000;
        }
        ans.getlength();
        return ans;
}

bool operator<(BigInt a,BigInt b)
{
        if(a.leng!=b.leng)return a.leng<b.leng;
        for(int i=a.leng-1;i>=0;i--)
                if(a[i]!=b[i])return a[i]<b[i];
        return false;
}
```

## 7.2 calculator

```
#define maxn 111
using namespace std;
struct node {
    int t; // t = 0 : num; t = 1 : operator.
    int value; // for op: +-*/()#^? == 012345678
    node (int _t, int _v) { t = _t, value = _v; }
    node () {}
} p[maxn];
char opt[] = "+-*/()#^?";
int omp[128];
int ask[13], asn;
int scan(char *str) {
    for (int i = 0; i < 9; ++i) omp[ opt[i] ] = i;
    int len = strlen(str);
    int cnt = 0, idx = 0, val;

    char op;
    asn = 0;
    for (idx = 0 ;idx < len; ) {
        if ( isdigit(str[idx]) ) {
            sscanf(str + idx, "%d", &val);
            p[cnt++] = node(0, val);
            while ( isdigit(str[idx]) ) ++idx;
        }
        else {
            sscanf(str + idx, "%c", &op);
            if (op == '?') ask[asn++] = cnt;
            p[cnt++] = node(1, omp[op]);
            ++idx;
        }
    }
    return cnt;
}

const int prior[8][8] = {
//    +   -   *   /   (   )   #   ^
    { 1,  1, -1, -1, -1,  1,  1, -1}, // +
    { 1,  1, -1, -1, -1,  1,  1, -1}, // -
    { 1,  1,  1,  1, -1,  1,  1, -1}, // *
    { 1,  1,  1,  1, -1,  1,  1, -1}, // /
    {-1, -1, -1, -1, -1,  0, -2, -1}, // (
    { 1,  1,  1,  1, -2,  1,  1,  1}, // )
    {-1, -1, -1, -1, -1, -2,  0, -1}, // #
    { 1,  1,  1,  1, -1,  1,  1,  1}  // ^
};


inline char chg(int c){
    char mp[] = "+-*/()#^";
    return mp[c];
}
struct Calculator{


    inline int atos(char* s){
        return atoi(s);
    }

    inline int operate(int a, int c, int b){
        switch (c) {
            case 0: return a + b;
            case 1: return a - b;
            case 2: return a * b;
            case 3: if(b == 0) return -inf;
                     else return a / b;
            default: return -1;
        }
    }

    int OPTR[maxn];
    int OPND[maxn];
```

```
    int calculate(int cnt){

        int lr = 0, ld = 0;
        OPTR[++lr] = 6;

        int idx = 0;

        int a, b, c;
        for (int i = 0; i < cnt; ++i) {
            if (p[i].t == 0) OPND[++ld] = p[i].value;
            else {
                switch (prior[OPTR[lr]][p[i].value]) {
                    case -1: OPTR[++lr] = p[i].value;
                             break;
                    case  0: lr--;
                             break;
                    case  1: c = OPTR[lr--];
                             b = OPND[ld--];
                             a = OPND[ld--];
                             //cout << lr << ":" << a << chg(c) << b <<endl;
                             OPND[++ld] = (operate(a, c, b));
                             if (OPND[ld] == -inf) return -inf;
                             --i;
                             break;
                }
            }
        }

        return OPND[ld];
    }
}cal;
```

## 7.3 Largest Submatrix of All 1's

```
int n, m;
bool mp[maxn][maxn];

int h[maxn][maxn];
int l[maxn], r[maxn];

int cal() {
    for (int i = 1; i <= n; ++i) {
        h[i][m + 1] = 0;
        for (int j = m; j >= 1; --j) {
            if (!mp[i][j]) h[i][j] = 0;
            else h[i][j] = h[i][j + 1] + 1;
        }
    }

    int ret = 0;
    int x1, y1, x2, y2;
    for (int j = 1; j <= m; ++j) {
        h[0][j] = h[n + 1][j] = -1;
        for (int i = 1; i <= n; ++i) {
            l[i] = i;
            while (h[l[i] - 1][j] >= h[i][j]) {
                l[i] = l[l[i] - 1];
            }
        }
        for (int i = n; i >= 1; --i) {
            r[i] = i;
            while (h[r[i] + 1][j] >= h[i][j]) {
                r[i] = r[r[i] + 1];
            }
        }
        for (int i = 1; i <= n; ++i) {
            x1 = l[i], x2 = r[i], y1 = j, y2 = j + h[i][j] - 1;
            ret = max(ret, (x2 - x1 + 1) * (y2 - y1 + 1));
        }
    }
    return ret;
}
```

## 7.4 xor from 1 to n

```
int xor_n(int n) {
    int t = n & 3;
    if (t & 1) return t / 2 ^ 1;
    return t / 2 ^ n;
}
```

## 7.5 (DP) Find kth number contains 666

```
#define inf 0x3f3f3f3f
#define Inf 0x3FFFFFFFFFFFFFFFLL
#define maxn 20
using namespace std;
typedef __int64 ll;
int num[maxn], m;
ll dp[maxn][4];

int dfs(int pos, int state, bool flag) {
    if(pos == -1) return state == 3;
    if(!flag && dp[pos][state] != -1) return dp[pos][state];
    int end = flag ? num[pos] : 9;
    ll ret = 0;
    for (int i = 0; i <= end; ++i) {
        if(state == 3) ret += dfs(pos - 1, 3, flag && i == end);
        else ret += dfs(pos - 1, (i == 6)? state + 1 : 0, flag && i == end);
    }
    if(!flag) dp[pos][state] = ret;
    return ret;
}
```

```
void init(ll n) {
    m = 0;
    for (; n; n /= 10) num[m++] = n % 10;
    num[m] = 0;
    memset(dp, 0xff, sizeof dp);
    dfs(m - 1, 0, true);
}

ll ans;

void find(int pos, int state, ll now, int k, bool flag) {
    if(pos == -1) {
        if(state == 3) ans = now;
        return;
    }
    int end = flag ? num[pos] : 9;
    int p, t;
    for (p = 0; p <= end; ++p) {
        if(state == 3) t = dfs(pos - 1, 3, flag && p == end);
        else t = dfs(pos - 1, (p == 6) ? state + 1 : 0, flag && p == end);
        if(t < k) k -= t;
        else break;
    }
    if(state == 3) find(pos - 1, 3, now * 10 + p, k, flag && p == end);
    else find(pos - 1, (p == 6) ? state + 1 : 0, now * 10 + p, k, flag && p
         == end);
}

int main(){
    init(10000000000LL);
    int T, k;
    for (scanf("%d", &T); T--; ) {
        scanf("%d", &k);
        find(m, 0, 0, k, true);
        printf("%I64d\n", ans);
    }
    return 0;
}
```

## 7.6 DLX

```
int const maxn = 1010;
int const maxi = maxn * maxn + maxn;
int U[maxi], D[maxi], L[maxi], R[maxi], C[maxi], W[maxi];
int S[maxn], O[maxn];
int n, m, K;

inline void remove(int c) {
    L[R[c]] = L[c];
    R[L[c]] = R[c];
    for (int i = D[c]; i != c; i = D[i]) {
        for (int j = R[i]; j != i; j = R[j]) {
            U[D[j]] = U[j];
            D[U[j]] = D[j];
            S[C[j]]--;
        }
    }
}
inline void resume(int c) {
    for (int i = U[c]; i != c; i = U[i]) {
        for (int j = L[i]; j != i; j = L[j]) {
            S[C[j]]++;
            U[D[j]] = D[U[j]] = j;
        }
    }
    L[R[c]] = R[L[c]] = c;
}
bool dfs() {
    if (R[0] == 0) return true;
    int s = inf, c;
    for (int t = R[0]; t != 0; t = R[t]) {
        if (S[t] < s) {
            s = S[t];
            c = t;
        }
    }
    remove(c);
    for (int i = D[c]; i != c; i = D[i]) {
        O[K] = W[i];
        for (int j = R[i]; j != i; j = R[j]) {
            remove(C[j]);
        }
        ++K;
        if (dfs()) return true;
        --K;
        for (int j = L[i]; j != i; j = L[j]) {
            resume(C[j]);
        }
    }
    resume(c);
    return false;
}
int mp[maxn][maxn], d[maxn];
int idx;

int main() {
    while (~scanf("%d%d", &n, &m)) {
        for (int i = 1; i <= n; ++i) {
            scanf("%d", &d[i]);
            for (int j = 0; j < d[i]; ++j) {
                scanf("%d", &mp[i][j]);
            }
            sort(mp[i], mp[i] + d[i]);
```

```
        }
        memset(S, 0, sizeof S);
        idx = 0;
        L[0] = m, R[0] = 1;
        for (int i = 1; i <= m; ++i) {
            L[i] = i - 1;
            R[i] = i + 1;
            U[i] = D[i] = i;
        }
        R[m] = 0;
        idx = m + 1;
        for (int i = 1; i <= n; ++i) {
            int s = idx, c;
            L[s] = R[s] = s;
            for (int j = 0; j < d[i]; ++j) {
                c = mp[i][j];
                S[c]++;
                W[idx] = i;
                C[idx] = c;
                U[idx] = U[c]; D[idx] = c; D[U[c]] = idx; U[c] = idx;
                L[idx] = L[s]; R[idx] = s; R[L[s]] = idx; L[s] = idx;
                ++idx;
            }
        }
        K = 0;
        bool flag = dfs();
        if (!flag) puts("NO");
        else {
            printf("%d␣", K);
            for (int i = 0; i < K; ++i) {
                if (i == K - 1) printf("%d\n", O[i]);
                else printf("%d␣", O[i]);
            }
        }

    }
    return 0;
}
```

## 7.7 vimrc

```
behave mswin
vnoremap <C-X> "+x
vnoremap <C-C> "+y
map <C-V> "+gP
cmap <C-V> <C-R>+
exe 'inoremap <script> <C-V>' paste#paste_cmd['i']
exe 'vnoremap <script> <C-V>' paste#paste_cmd['v']
noremap <C-S> :update<CR>
inoremap <C-S> <C-O>:update<CR>
noremap <C-A> gggH<C-O>G
inoremap <C-A> <C-O>gg<C-O>gH<C-O>G
inoremap <C-D> <C-O>dd
noremap <C-Z> u
inoremap <C-Z> <C-O>u
map <F3> 0i//<C-C>
map <F4> ^xx
inoremap <CR> <CR><space><bs>
nnoremap o o<space><bs>
nnoremap O O<space><bs>
noremap <F6> =a{
inoremap { {<c-c>==+?{<cr>a
inoremap } }<c-c>==+?}<cr>a
au GUIEnter * simalt ~x
cd F:\vim
syn on
colo torte
se lines=40 co=130 cb+=unnamed nu sw=4 ts=4 nobk cin nocp mouse=a bs=2
    hi=50 gfn=Courier_New:h12:cANSI
map <c-t> :tabnew<CR>
map <tab> :tabn<CR>
map <s-tab> :tabp<CR>
map <c-w> :close<cr>
inoremap <F10> <C-C>:call CR()<CR>
map <F10> :call CR()<CR>
func CR()
exec "w"
exec "!start cmd /c g++ %<.cc -o %<.exe & %<.exe < %<.in & pause"
endfunc

inoremap <F9> <C-C>:call CR2()<CR>
map <F9> :call CR2()<CR>
func CR2()
exec "w"
exec "!start cmd /c g++ %<.cc -o %<.exe & %<.exe & pause"
endfunc

inoremap <F2> <C-C>:call CR3()<CR>
map <F2> :call CR3()<CR>
func CR3()
exec "vsplit"
exec "vi %<.in"
endfunc

inoremap <F5> <C-C>:call SetTitle()<CR> GkkO
map <F5> :call SetTitle()<CR> GkkO
func SetTitle()
call setline(1, "#include <iostream>")
call append(line("."), "#include <cstdio>")
call append(line(".")+1, "#include <cstdlib>")
call append(line(".")+2, "#include <cstring>")
call append(line(".")+3, "#include <algorithm>")
call append(line(".")+4, "#include <cmath>")
call append(line(".")+5, "#include <string>")
```

```
call append(line(".")+6, "#include <vector>")
call append(line(".")+7, "#include <queue>")
call append(line(".")+8, "#include <set>")
call append(line(".")+9, "#include <map>")
call append(line(".")+10, "#include <ctime>")
call append(line(".")+11, "")
call append(line(".")+12, "#define inf 0x3f3f3f3f")
call append(line(".")+13, "#define Inf 0x3FFFFFFFFFFFFFFFLL")
call append(line(".")+14, "using namespace std;")
call append(line(".")+15, "")
call append(line(".")+16, "int main() {")
"call append(line(".")+17, " freopen(\"".expand("%<:t").".in\", \"r\",
    stdin);")
call append(line(".")+17, " return 0;")
call append(line(".")+18, "}")
call append(line(".")+19, "")
endfunc



nmap <C-F> <Esc>:Setcomment<CR>
imap <C-F> <Esc>:Setcomment<CR>
vmap <C-F> <Esc>:SetcommentV<CR>
command! -nargs=0 Setcomment call s:SET_COMMENT()
command! -nargs=0 SetcommentV call s:SET_COMMENTV()

"non visual
function! s:SET_COMMENT()
    let lindex=line(".")
    let str=getline(lindex)
    let CommentMsg=s:IsComment(str)
    call s:SET_COMMENTV_LINE(lindex,CommentMsg[1],CommentMsg[0])
endfunction

"visual
function! s:SET_COMMENTV()
    let lbeginindex=line("'<")
    let lendindex=line("'>")
    let i=lbeginindex
    while i<=lendindex
    let str=getline(i)
    let CommentMsg=s:IsComment(str)
        call s:SET_COMMENTV_LINE(i,CommentMsg[1],CommentMsg[0])
        let i=i+1
    endwhile
endfunction

function! s:SET_COMMENTV_LINE( index,pos, comment_flag )
    let poscur = [0, 0,0, 0]
    let poscur[1]=a:index
    let poscur[2]=a:pos+1
    call setpos(".",poscur)
    if a:comment_flag==0
        exec "normal! i//
    else
        exec "normal! xx"
    endif
endfunction

function! s:IsComment(str)
    let ret= [0, 0]
    let i=0
    let strlen=len(a:str)
    while i<strlen
        if !(a:str[i]==' ' || a:str[i] == ' ' )
            let ret[1]=i
            if a:str[i]=='/' && a:str[i+1]=='/'
                let ret[0]=1
            else
                let ret[0]=0
            endif
            return ret
        endif
        let i=i+1
    endwhile
    return [0,0]
endfunction


"set guifont=Consolas\ 12

inoremap ( ()<Esc>i
inoremap [ []<Esc>i
inoremap { {<CR>}<Esc>O
autocmd Syntax html,vim inoremap < <lt>><Esc>i| inoremap >
     <c-r>=ClosePair('>')<CR>
inoremap ) <c-r>=ClosePair(')')<CR>
inoremap ] <c-r>=ClosePair(']')<CR>
inoremap } <c-r>=CloseBracket()<CR>
inoremap " <c-r>=QuoteDelim('"')<CR>
inoremap ' <c-r>=QuoteDelim("'")<CR>

function ClosePair(char)
 if getline('.')[col('.') - 1] == a:char
 return "\<Right>"
 else
 return a:char
 endif
endf

function CloseBracket()
 if match(getline(line('.') + 1), '\s*}') < 0
 return "\<CR>}"
```

```
 else
 return "\<Esc>j0f}a"
 endif
endf

function QuoteDelim(char)
 let line = getline('.')
 let col = col('.')
 if line[col - 2] == "\\"
 "Inserting a quoted quotation mark into the string
 return a:char
 elseif line[col - 1] == a:char
 "Escaping out of the string
 return "\<Right>"
 else
 "Starting a string
 return a:char.a:char."\<Esc>i"
 endif
endf

colors vividchalk

if has('gui_running')
  set guifont=Consolas:h11
endif

set ofu=syntaxcomplete#Complete
imap <silent> ‘ <C-X><C-O>
```