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

#define tpl_ template
#define tn_ typename
#define op_ operator
#define cx_ constexpr
#define fn function
#define f(i, to) for (int i = 0; i < to; ++i)
#define fe(i, to) for (int i = 1; i <= to; ++i)
#define rep(i, a, b) for (int i = a; i <= b; ++i)
#define repr(i, a, b) for (int i = a; i >= b; --i)
#define ff first
#define ss second
#define pb push_back
#define fora(a, x) for (auto &(a) : (x))
#define all(x) begin(x), end(x)
#define rall(x) rbegin(x), rend(x)
#define print(x) (cout << #x << "=" << (x) << endl)
#define quit(s) do { cout << (s) << en; return; } while (false)

#define int long long
tpl_<tn_ T> using v = vector<T>;
using vi = v<int>;
tpl_<tn_ T> using vv = v<v<T>>;
using ll = long long;
using pii = pair<int, int>;
using vb = v<bool>;
using vvb = v<vb>;
using vs = v<string>;
using iii = array<int, 3>;
using i4 = array<int, 4>;
using vvi = v<vi>;
using vll = v<ll>;
using vpii = v<pii>;
using vvpii = v<vpii>;
tpl_<tn_ K, tn_ T> using ump = unordered_map<K, T>;
tpl_<tn_ T> using ust = unordered_set<T>;
tpl_<tn_ T> using mset = multiset<T>;
tpl_<tn_ T> using pq = priority_queue<T>;
tpl_<tn_ T> using mpq = priority_queue<T, v<T>, greater<T>>;
tpl_<tn_ It, tn_ T> auto leq_bound(It l, It r, T x) {
  auto it = upper_bound(l, r, x);
  return it != l ? prev(it) : r;
}
tpl_<tn_ It, tn_ T> auto less_bound(It l, It r, T x) {
  auto it = lower_bound(l, r, x);
  return it != l ? prev(it) : r;
}
```

```
tpl_<tn_ T, tn_ U> T fstTrue(T l, T r, U ff) {
  for (++r; l < r;) {
    T m = l + (r - l) / 2;
    if (ff(m))
      r = m;
    else
      l = m + 1;
  }
  return l;
}
tpl_<tn_ T, tn_ U> T lstTrue(T l, T r, U ff) {
  for (++r; l < r;) {
    T m = l + (r - l) / 2;
    if (ff(m))
      l = m + 1;
    else
      r = m;
  }
  return l - 1;
}
tpl_<tn_ T> bool cmn(T &a, T b) { return b < a ? a = b, 1 : 0; }
tpl_<tn_ T> bool cmx(T &a, T b) { return a < b ? a = b, 1 : 0; }
cx_ auto en = "\n";
cx_ auto sp = " ";
using str = string;
typedef fn<void(int, int)> fvii;
typedef fn<void(int, int, int)> fviii;
cx_ int INF = 1e9;
cx_ ll INFL = 0x3f3f3f3f3f3f3f3f;
cx_ int B = 31;

void setIO(const str &name = "") {
  ios_base::sync_with_stdio(false);
  cin.tie(nullptr);
  if (!name.empty()) {
    freopen((name + ".in").c_str(), "r", stdin);
    freopen((name + ".out").c_str(), "w", stdout);
  }
}
tpl_<tn_ A, tn_ B> ostream &op_ << (ostream & os, const pair<A, B> &p) {
  return os << "(" << p.ff << ", " << p.ss << ")";
}
tpl_<tn_ A> ostream &op_ << (ostream & o, const v<v<A>> &m) {
  for (auto &r : m) {
    o << "{";
    for (auto &e : r)
      o << e << " ";
    o << "}\n";
  }
  return o;
}
tpl_<tn_ K, tn_ T> ostream &op_ << (ostream & o, const map<K, T> &m) {
  o << "{";
  for (auto &p : m)
    o << p.ff << ":" << p.ss << ", ";
  return o << "}";
```

```
}
tpl_<tn_ C, tn_ T = enable_if_t<!is_same_v<C, str>, tn_ C::value_type>> ostream
        &op_
    << (ostream & os, const C &v) {
  for (const T &x : v)
    os << ' ' << x;
  return os;
}
struct cind {
  tpl_<tn_ T> cind &op_ >> (T & x) {
    cin >> x;
    --x;
    return *this;
  }
} cind;
struct bout {
  tpl_<tn_ T> bout &op_ << (T x) {
    if cx_ (is_integral_v<T>) {
      int y = x;
      if (y == 0) { cout << '0'; return *this; }
      if (y < 0) { cout << '-'; y = -y; }
      str s;
      while (y) { s.pb('0' + (y & 1)); y >>= 1; }
      reverse(all(s));
      cout << s;
    } else cout << x;
    return *this;
  }
} bout;
void read(vi &v) { for (auto &x : v) cin >> x; }
void read(vpii &v) { for (auto &[x, y] : v) cin >> x >> y; }
void read(vvi &a) { for (auto &r : a) for (auto &x : r) cin >> x; }
void read(vvi &g, int m, bool o = true, bool d = false) {
  f(i, m) {
    int u, v;
    cin >> u >> v;
    if (o) { u--; v--; }
    g[u].pb(v);
    if (!d) g[v].pb(u);
  }
}
void read(vvpii &g, int m, bool dec = true, bool dir = false) {
  f(i, m) {
    int u, v, w;
    cin >> u >> v >> w;
    if (dec) { u--; v--; }
    g[u].pb({v, w});
    if (!dir) g[v].pb({u, w});
  }
}
```

```
struct DSU {
  vi p, sz;
  explicit DSU(const int n) { p.resize(n), sz.resize(n, 1), iota(all(p), 0); }
  int par(int x) { return x == p[x] ? x : p[x] = par(p[x]); }
  void merge(int x, int y) {
    x = par(x), y = par(y);
    if (x != y) {
      if (sz[x] < sz[y]) swap(x, y);
      p[y] = x, sz[x] += sz[y];
    }
  }
  bool same(int x, int y) { return par(x) == par(y); }
};

tpl_<tn_ T, tn_ C> struct Segtree {
  int n, N;
  v<T> t, nums;
  C c;
  T z;
  static_assert(is_invocable_r_v<T, C, T, T>, "Combine must be T(T,T)");
  Segtree(int sz, C c, const v<T> &init = {}, const T &z = T())
      : n(sz), N(1), nums(sz, z), c(std::move(c)), z(z) {
    while (N < n)
      N <<= 1;
    t.assign(2 * N, z);
    if (!init.empty()) {
      nums = init;
      for (int i = 0; i < n; i++)
        t[N + i] = nums[i];
    }
    for (int i = N - 1; i; i--)
      t[i] = c(t[i << 1], t[i << 1 | 1]);
  }
  void add(int p, T x) {
    int i = p + N;
    t[i] = c(t[i], x);
    for (i >>= 1; i; i >>= 1)
      t[i] = c(t[i << 1], t[i << 1 | 1]);
  }
  void update(int p, T x) {
    int i = p + N;
    t[i] = x;
    for (i >>= 1; i; i >>= 1)
      t[i] = c(t[i << 1], t[i << 1 | 1]);
  }
  T query(int l, int r) {
    if (r < l)
      return z;
    l += N;
    r += N;
    T L = z, R = z;
    while (l <= r) {
      if (l & 1)
        L = c(L, t[l++]);
      if (!(r & 1))
```

```
        R = c(t[r--], R);
      l >>= 1;
      r >>= 1;
    }
    return c(L, R);
  }
};

tpl_<tn_ T, tn_ U, tn_ C, tn_ Ap,
     tn_ Cmp> struct LazySegtree { // NOTE this "apply" passes in [l, r] by
                                   // default! ap and cmp use {old, new op ... }
  static_assert(is_invocable_r_v<T, C, T, T>, "Combine T(T,T)");
  static_assert(is_invocable_r_v<T, Ap, T, U, int, int>,
                "Apply T(T,U,int,int)");
  static_assert(is_invocable_r_v<U, Cmp, U, U>, "Compose U(U,U)");
  int n;
  T z;
  v<T> t, nums;
  v<U> ops;
  C c;
  Ap ap;
  Cmp cmp;
  LazySegtree(int sz, C cmb, Ap _ap, Cmp _cmp, const v<T> &i = {},
              const T &z = T())
      : n(sz), t(4 * sz), nums(sz), ops(4 * sz), c(move(cmb)), ap(move(_ap)),
        cmp(move(_cmp)) {
    if (!i.empty()) {
      nums = i;
      build(1, 0, n - 1);
    }
  }
  void add(int l, int r, U u) { add(1, 0, n - 1, l, r, u); }
  T query(int l, int r) { return query(1, 0, n - 1, l, r); }

private:
  void build(int i, int a, int b) {
    if (a == b) {
      t[i] = nums[a];
      return;
    }
    int m = (a + b) / 2;
    build(2 * i, a, m);
    build(2 * i + 1, m + 1, b);
    t[i] = c(t[2 * i], t[2 * i + 1]);
  }
  void applyNode(int i, const U &u, int a, int b) {
    t[i] = ap(t[i], u, a, b);
    ops[i] = cmp(ops[i], u);
  }
  void push(int i, int a, int b) {
    int m = (a + b) / 2;
    applyNode(2 * i, ops[i], a, m);
    applyNode(2 * i + 1, ops[i], m + 1, b);
    ops[i] = U();
  }
  void add(int i, int a, int b, int l, int r, const U &u) {
```

```
    if (r < a || b < l)
      return;
    if (l <= a && b <= r) {
      applyNode(i, u, a, b);
      return;
    }
    push(i, a, b);
    int m = (a + b) / 2;
    add(2 * i, a, m, l, r, u);
    add(2 * i + 1, m + 1, b, l, r, u);
    t[i] = c(t[2 * i], t[2 * i + 1]);
  }
  T query(int i, int a, int b, int l, int r) {
    if (r < a || b < l)
      return z;
    if (l <= a && b <= r)
      return t[i];
    push(i, a, b);
    int m = (a + b) / 2;
    T L = query(2 * i, a, m, l, r), R = query(2 * i + 1, m + 1, b, l, r);
    return c(L, R);
  }
};

tpl_<tn_ T, tn_ C> struct BIT {
  int n;
  v<T> t, nums;
  C c;
  static_assert(is_invocable_r_v<T, C, T, T>, "Combine must be T(T,T)");
  BIT(int sz, C c, const v<T> &init = {}, T z = T())
      : n(sz), t(sz + 1, z), nums(sz, z), c(move(c)) {
    if (!init.empty()) {
      f(i, n) add(i, init[i]);
    }
  }
  void add(int i, T x) {
    nums[i] += x;
    for (i += 1; i <= n; i += (i & -i))
      t[i] = c(t[i], x);
  }
  void update(int i, T x) {
    T diff = x - nums[i];
    nums[i] = x;
    for (i += 1; i <= n; i += (i & -i))
      t[i] = c(t[i], diff);
  }
  T query(int i) {
    T res = T();
    for (i += 1; i > 0; i -= (i & -i))
      res = c(res, t[i]);
    return res;
  }
  T query(int l, int r) { return query(r) - query(l - 1); }
};

tpl_<tn_ T> void printSegtree(int n, v<T> &t, int r = 20, int l = 4) {
```

```
  [[maybe_unused]] int rows = 0;
  fn<void(int, int, int, int)> dfs = [&](int i, int L, int R, int d) {
    if (L > R || rows >= r || d > l)
      return;
    rows++;
    cout << str(d * 2, ' ') << "[" << L << "," << R << "]: " << t[i] << en;
    if (L < R) {
      int M = (L + R) / 2;
      dfs(2 * i, L, M, d + 1);
      dfs(2 * i + 1, M + 1, R, d + 1);
    }
  };
  cout << "Segtree\n";
  dfs(1, 0, n - 1, 0);
}
tpl_<tn_ T, tn_ C> void printBIT(const BIT<T, C> &b, int l = 16) {
  cout << "BIT:\n";
  int lv = 0;
  while (1 << lv <= min(b.n, l))
    lv++;
  int c = min(b.n, l);
  v<vs> g(lv, vs(c, str(4, ' ')));
  fe(i, c) {
    int r = __builtin_ctz(i);
    if (r < lv)
      g[r][i - 1] = format("{:4}", b.t[i]);
  }
  f(r, lv) {
    f(c2, c) cout << g[r][c2];
    cout << en;
  }
}

struct Line {
  mutable int a, b, p; // =ax+b, last optimal x
  Line(int a, int b, int p = 0) : a(a), b(b), p(p) {}
  int at(int x) const { return a * x + b; }
  bool operator<(const Line &o) const { return a < o.a; }
  bool operator<(int x) const { return p < x; }
  friend long double intersect(Line x, Line y) {
    return static_cast<long double>(y.b - x.b) /
           static_cast<long double>(x.a - y.a);
  }
};

// For all CHT, default is upper hull / query max. For query min, negate line a,
// b, and query result.
struct CHT : multiset<Line, less<>> {
  static constexpr int inf = LLONG_MAX;
  static int floor(int a, int b) { return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator it1, iterator it2) {
    if (it2 == end()) {
      it1->p = inf;
      return false;
    }
    if (it1->a == it2->a)
```

```
      it1->p = it1->b > it2->b ? inf : -inf;
    else
      it1->p = floor(it2->b - it1->b, it1->a - it2->a);
    return it1->p >= it2->p;
  }
  void add(int a, int b) {
    auto z = insert({a, b, 0});
    auto y = z++, x = y;
    while (isect(y, z))
      z = erase(z);
    if (x != begin() && isect(--x, y))
      isect(x, y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y));
  }
  int query(int x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.a * x + l.b;
  }
};

tpl_<bool upperHull = true> struct MonotonicCHT {
  deque<Line> h;
  static bool badUpper(const Line &x, const Line &y, const Line &z) {
    ll lhs = (y.b - x.b) * (y.a - z.a), rhs = (z.b - y.b) * (x.a - y.a);
    return lhs >= rhs;
  }
  void add(const Line &ln) {
    if (!upperHull) {
      ln.a = -ln.a;
      ln.b = -ln.b;
    }
    while (h.size() >= 2 && badUpper(h[h.size() - 2], h.back(), ln))
      h.pop_back();
    h.pb(ln);
  }
  int query(int x) {
    if (h.empty())
      return upperHull ? -INFL : INFL;
    while (h.size() >= 2) {
      int v0 = h[0].at(x), v1 = h[1].at(x);
      if (v1 >= v0)
        h.pop_front();
      else
        break;
    }
    int res = h.front().at(x);
    return upperHull ? res : -res;
  }
};

void _dijkstra(vi &d, vvpii &adj, int a = 0) {
  mpq<pii> q;
  d[a] = 0, q.push({0, a});
  while (!q.empty()) {
```

```
    auto [w, u] = q.top();
    q.pop();
    if (w != d[u])
      continue;
    for (auto [v, dw] : adj[u]) {
      if (w + dw < d[v]) {
        d[v] = w + dw;
        q.push({d[v], v});
      }
    }
  }
}

tuple<vi, vi, vi, vi, vi> _dfs(const vvi &g, int a = 0) {
  int n = g.size(), t = 0;
  vi sz(n, 1), par(n, -1), dep(n, 0), in(n, 0), out(n, 0);
  fviii dfs = [&](int u, int p, int d) {
    par[u] = p;
    dep[u] = d;
    in[u] = t++;
    for (int v : g[u])
      if (v != p) {
        dfs(v, u, d + 1);
        sz[u] += sz[v];
      }
    out[u] = t++;
  };
  dfs(a, -1, 0);
  return {sz, dep, par, in, out};
}
tuple<vi, vi, vi, vi> _dfs(vvpii &g, int a = 0) {
  int n = g.size();
  vi sz(n, 1), par(n, -1), dep(n, 0), dist(n, 0);
  fviii dfs = [&](int u, int p, int d) {
    par[u] = p;
    dep[u] = d;
    for (auto [v, w] : g[u])
      if (v != p) {
        dist[v] = w;
        dfs(v, u, d + 1);
        sz[u] += sz[v];
      }
  };
  dfs(a, -1, 0);
  return {sz, dep, par, dist};
}

vvi _jump(const vi &par, int out = -1) {
  int n = par.size();
  int ln = log2(n) + 1;
  vvi up(n, vi(ln, 0));
  f(i, n) up[i][0] = par[i];
  rep(j, 1, ln - 1) {
    f(i, n) {
      int p = up[i][j - 1];
      if (p == out)
```

```
        up[i][j] = out;
      else
        up[i][j] = up[p][j - 1];
    }
  }
  return up;
}

tpl_<tn_ F> pair<vvi, vvi> _jumpW(vi &par, int out, vi &wt, F mrg) {
  int n = par.size(), ln = log2(n) + 1;
  vvi up(n, vi(ln, 0)), c(n, vi(ln, 0));
  f(i, n) {
    up[i][0] = par[i];
    c[i][0] = (par[i] == out ? 0 : wt[i]);
  }
  rep(j, 1, ln - 1) {
    f(i, n) {
      int p = up[i][j - 1];
      if (p == out) {
        up[i][j] = out;
        c[i][j] = c[i][j - 1];
      } else {
        up[i][j] = up[p][j - 1];
        c[i][j] = mrg(c[i][j - 1], c[p][j - 1]);
      }
    }
  }
  return {up, c};
}

int _lca(int u, int v, const vvi &up, const vi &d) {
  int ln = log2(up.size()) + 1;
  if (d[u] < d[v])
    swap(u, v);
  rep(j, 0, ln - 1) {
    if (d[u] - d[v] & (1 << j))
      u = up[u][j];
  }
  if (u == v)
    return u;
  repr(j, ln - 1, 0) {
    if (up[u][j] != up[v][j]) {
      u = up[u][j], v = up[v][j];
    }
  }
  return up[u][0];
}

pair<map<int, int>, vi> _compress(vi &a) {
  vi v = a;
  sort(all(v));
  v.erase(unique(all(v)), v.end());
  map<int, int> mp;
  auto it = mp.end();
  f(i, v.size()) it = mp.emplace_hint(it, v[i], i);
  for (int &x : a)
```

```
    x = mp[x];
  return {mp, v};
}

vi _virtualTree(vvi &vadj, const vi &nodes, const vi &tin, const vvi &up,
                const vi &d, bool dir = true) {
  vi lu = nodes;
  auto cmp = [&](int u, int v) { return tin[u] < tin[v]; };
  int n = lu.size();
  lu.reserve(2 * n);
  sort(all(lu), cmp);
  f(i, n - 1) lu.pb(_lca(lu[i], lu[i + 1], up, d));
  sort(all(lu), cmp);
  lu.erase(unique(all(lu)), lu.end());
  for (int u : lu)
    vadj[u].clear();
  f(i, lu.size() - 1) {
    int u = _lca(lu[i], lu[i + 1], up, d);
    int v = lu[i + 1];
    vadj[u].pb(v);
    if (!dir) vadj[v].pb(u);
  }
  return lu;
}

struct pairHash {
  tpl_<tn_ A, tn_ B> size_t op_()(const pair<A, B> &p) const {
    return hash<A>{}(p.ff) ^ (hash<B>{}(p.ss) << 1);
  }
};
struct vHash {
  tpl_<tn_ T> size_t op_()(const v<T> &x) const {
    size_t h = 0;
    for (auto &i : x)
      h ^= hash<T>{}(i) + 0x9e3779b9 + (h << 6) + (h >> 2);
    return h;
  }
};

auto _add = [](int a, int b) { return a + b; };
auto _sub = [](int a, int b) { return a - b; };
auto _sortinv = [](const pii &a, const pii &b) {
  if (a.ff == b.ff)
    return a.ss > b.ss;
  return a.ff < b.ff;
};
vpii dirs = {{1, 0}, {0, -1}, {0, 1}, {-1, 0}};
map<char, int> dirMap = {{'E', 0}, {'S', 1}, {'N', 2}, {'W', 3}};
auto check = [](auto y, auto x, auto m, auto n) {
  return y >= 0 && y < m && x >= 0 && x < n;
};

cx_ int N = 100000;
cx_ int MOD = 1e9 + 7; // 998244353;
inline int add(int a, int b) {
  int s = a + b;
```

```
  return s < MOD ? s : s - MOD;
}
inline int sub(int a, int b) {
  int s = a - b;
  return s >= 0 ? s : s + MOD;
}
inline int ceil(int a, int b) { return a >= 0 ? (a + b - 1) / b : a / b; }
inline int mult(int a, int b) { return a * b % MOD; }

inline int fpow(int a, int b) {
  int res = 1;
  a %= MOD;
  while (b > 0) {
    if (b & 1)
      res = res * a % MOD;
    a = mult(a, a);
    b >>= 1;
  }
  return res;
}
inline int inv(int x) { return fpow(x, MOD - 2); }

struct mint {
  ll v;
  mint(ll x = 0) : v((x % MOD + MOD) % MOD) {}
  mint op_ + (mint o) const { return mint(add(v, o.v)); }
  mint op_ - (mint o) const { return mint(sub(v, o.v)); }
  mint op_ *(mint o) const { return mint(mult(v, o.v)); }
  mint op_ / (mint o) const { return mint(mult(v, inv(o.v))); }
  mint &op_ += (mint o) { return *this = *this + o; }
  mint &op_ -= (mint o) { return *this = *this - o; }
  mint &op_ *= (mint o) { return *this = *this * o; }
  mint &op_ /= (mint o) { return *this = *this / o; }
  friend ostream &op_ << (ostream & os, const mint &x) { return os << x.v; }
};

cx_ ll msb(ll x) { return 63 - __builtin_clzll(x); }
cx_ ll lsb(ll x) { return __builtin_ctz(x); }

vb sieve(int n) {
  vb p(n + 1, true);
  p[0] = p[1] = false;
  for (int i = 2; i * i <= n; ++i)
    if (p[i])
      for (int j = i * i; j <= n; j += i)
        p[j] = false;
  return p;
}

vi sieveList(int n) {
  vb p = sieve(n);
  vi primes;
  rep(i, 2, n) if (p[i]) primes.pb(i);
  return primes;
}
```

```
vi sieveSPF(int n) {
  vi spf(n + 1);
  fe(i, n) spf[i] = i;
  for (int i = 2; i * i <= n; ++i) {
    if (spf[i] != i)
      continue;
    for (int j = i * i; j <= n; j += i) {
      if (spf[j] == j)
        spf[j] = i;
    }
  }
  return spf;
}

pair<vi, vi> initFact(int n) {
  vi fa(n + 1), ifa(n + 1);
  fa[0] = 1;
  fe(i, n) fa[i] = mult(fa[i - 1], i);
  ifa[n] = inv(fa[n]);
  repr(i, n - 1, 0) ifa[i] = mult(ifa[i + 1], i + 1);
  return {fa, ifa};
}

class Matrix {
public:
  vvi v;
  explicit Matrix(int n) : v(n, vi(n, 0)) {}
  Matrix op_ *(const Matrix &m) const {
    int n = v.size();
    Matrix r(n);
    f(i, n) f(k, n) f(j, n) r.v[i][j] = (r.v[i][j] + v[i][k] * m.v[k][j]) % MOD;
    return r;
  }
  Matrix op_ ^ (ll p) const {
    int n = v.size();
    Matrix r(n), b = *this;
    f(i, n) r.v[i][i] = 1;
    while (p) {
      if (p & 1)
        r = r * b;
      b = b * b;
      p >>= 1;
    }
    return r;
  }
};

int k, n, m;
void solve() {}

int32_t main() {
  setIO();
  // int t; cin>>t; f(i, t) solve();
}
```