Олимпиада СПбГУ
по информатике
2023/24 учебного года

24fin0222 – Конобейцев Иван Олегович

A	B	C	D	E	F	sum
30	50	100	100	20	67	367

Task A

```
// Online C++ compiler to run C++ program online
#pragma GCC optimize(O3)
#pragma comment(linker, "/stack:2000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
```

```
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <queue>
#include <cmath>
#include <stack>
#include <string>
#include <map>
#include <unordered_map>
#include <set>
#include <random>
#include <chrono>
#include <deque>
#include <string>
#include <bitset>

using namespace std;

//#define int long long
int inf = 1000000000;
int n, a, b;
set<int> ans;
```

```
int u;

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

vector<int> st(90);
int mult = 29;


int solve(int a1, int b1, string s) {
  if (a1 == 0 and b1 == 0) {
    for (int i = 0; i < u; i++) {
      int u2 = 0;
      int now = 0;
      set<int> s2;
      for (int j2 = 0; j2 < s.size(); j2++) {
        char j = s[j2];
        if (j == '0') {
          now += a;
        }
        else now += b;
        u2 += st[(now + i) % n];
      }
      ans.insert(u2);
    }
  }
  int ha = 0;
  if (a1 > 0) ha += solve(a1 - 1, b1, s + '0');
  if (b1 > 0) ha += solve(a1, b1 - 1, s + '1');
  return ha;
}


signed main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);



  cin >> a >> b >> n;

  u = a * b / gcd(a, b);
  st[0] = 1;
  for (int i = 1; i < 90; i++) {
    st[i] = st[i - 1] * mult;
  }

  for (int i = 0; i < 80; i++) {
    if (n - i * a < 0 or (n - i * a) % b != 0) continue;
    int a1 = i;
```

```
    int b1 = (n - i * a) / b;
    //cout << a1 << " " << b1 << " lol\n";
    int u = solve(a1, b1, "");

  }

  cout << ans.size();


}
```

Task B

```
// Online C++ compiler to run C++ program online
#pragma GCC optimize(O3)
#pragma comment(linker, "/stack:2000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
```

```
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <queue>
#include <cmath>
#include <stack>
#include <string>
#include <map>
#include <unordered_map>
#include <set>
#include <random>
#include <chrono>
#include <deque>
#include <string>
#include <bitset>

using namespace std;

#define int long long

int n;


signed main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  cin >> n;
  int a1, b1, c1;
  cin >> a1 >> b1 >> c1;
  int a2, b2, c2;
  cin >> a2 >> b2 >> c2;
```

```
  int u = abs(a1 - a2) + abs(b1 - b2) + abs(c1 - c2);
  cout << u / 2;


}
```

Task C

```
// Online C++ compiler to run C++ program online
#pragma GCC optimize(O3)
#pragma comment(linker, "/stack:2000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
```

```
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <queue>
#include <cmath>
#include <stack>
#include <string>
#include <map>
#include <unordered_map>
#include <set>
#include <random>
#include <chrono>
#include <deque>
#include <string>
#include <bitset>

using namespace std;

#define int long long

int n;
int mult = 91;
int mod1 = 1000000007;
int mod2 = 2286661337;

struct hash1 {
  int val, mod;
  hash1 operator + (hash1 a) {
    return { (this->val + a.val) % a.mod, a.mod };
  }
  hash1 operator * (hash1 a) {
    return { (this->val * a.val) % a.mod, a.mod };
  }
  hash1 operator - (hash1 a) {
    return { (this->val - a.val + a.mod) % a.mod, a.mod };
  }
  bool operator == (hash1 a) {
    return this->val == a.val;
```

```
  }
  bool operator > (hash1 a) {
    return this->val > a.val;
  }
  bool operator < (hash1 a) {
    return this->val < a.val;
  }
};

int best = 0;
string s2;
vector<hash1> l1;
vector<hash1> l2;
vector<hash1> st1(200000, { 1, mod1 });
vector<hash1> st2(200000, { 1, mod2 });

bool isSame(int left1, int right1, int left2, int right2) {
  left1++;
  right1++;
  left2++;
  right2++;
  if (left1 > right1) return true;
  if (left1 > left2) swap(left1, left2);
  hash1 a = (l1[right1] - l1[left1 - 1]) * st1[left2 - left1];
  hash1 b = (l1[right2] - l1[left2 - 1]);
  hash1 a2 = (l2[right1] - l2[left1 - 1]) * st2[left2 - left1];
  hash1 b2 = (l2[right2] - l2[left2 - 1]);
  return a == b and a2 == b2;
}

char comp(int left1, int right1, int left2, int right2) {
  int left = -1;
  int right = right1 - left1 + 1;
  int mid;
  while (left < right - 1) {
    mid = (left + right) / 2;
    if (isSame(left1, left1 + mid - 1, left2, left2 + mid - 1)) {
      left = mid;
    }
    else {
      right = mid;
    }
  }
  //cout << left1 << " " << right1 << " " << left2 << " " << right2 << "
    " << left << " ehhee\n";
  if (s2[left1 + left] == s2[left2 + left]) return '=';
  if (s2[left1 + left] > s2[left2 + left]) return '>';
  if (s2[left1 + left] < s2[left2 + left]) return '<';
}


signed main() {
  ios::sync_with_stdio(false);
```

```
cin.tie(0);
cout.tie(0);

for (int i = 1; i < 200000; i++) {
  st1[i] = st1[i - 1] * hash1{ mult, mod1 };
}
for (int i = 1; i < 200000; i++) {
  st2[i] = st2[i - 1] * hash1{ mult, mod2 };
}


string type;
cin >> type;
int n;
cin >> n;
string s;
cin >> s;
if (type == "first") {
  s2 = s + s;
  l1 = vector<hash1>(n * 2 + 1, {0, mod1});
  l2 = vector<hash1>(n * 2 + 1, {0, mod2});
  for (int i = 1; i < n * 2 + 1; i++) {
    l1[i] = l1[i - 1] + hash1{ s2[i - 1] - '0', mod1 } * st1[i - 1];
    l2[i] = l2[i - 1] + hash1{ s2[i - 1] - '0', mod2 } * st2[i - 1];
  }
  for (int i = 1; i < n; i++) {
    //cout << best << " " << comp(best, best + n - 1, i, i + n - 1) <<
 " ehjhee\n";
    if (comp(best, best + n - 1, i, i + n - 1) == '<') {
      best = i;
    }
  }
  hash1 u = { 0, mod1 };
  hash1 u2 = { 0, mod2 };
  for (int i = best; i < best + n; i++) {
    u = u + hash1{ s2[i] - '0', mod1 } * st1[i - best];
    u2 = u2 + hash1{ s2[i] - '0', mod2 } * st2[i - best];
  }
  string ans = "1337";
  for (int t = 0; t < 5; t++) {
    ans += char(u.val % mult + 33);
    u.val = u.val / mult;
  }
  for (int t = 0; t < 5; t++) {
    ans += char(u2.val % mult + 33);
    u2.val = u2.val / mult;
  }
  cout << ans;

}
else {
  s2 = s + s;
  l1 = vector<hash1>(n * 2 + 1, { 0, mod1 });
```

```
    l2 = vector<hash1>(n * 2 + 1, { 0, mod2 });
    for (int i = 1; i < n * 2 + 1; i++) {
      l1[i] = l1[i - 1] + hash1{ s2[i - 1] - '0', mod1 } *st1[i - 1];
      l2[i] = l2[i - 1] + hash1{ s2[i - 1] - '0', mod2 } *st2[i - 1];
    }
    for (int i = 1; i < n; i++) {
      //cout << best << " " << comp(best, best + n - 1, i, i + n - 1) <<
    " ehjhee\n";
      if (comp(best, best + n - 1, i, i + n - 1) == '<') {
        best = i;
      }
    }
    hash1 u = { 0, mod1 };
    hash1 u2 = { 0, mod2 };
    for (int i = best; i < best + n; i++) {
      u = u + hash1{ s2[i] - '0', mod1 } *st1[i - best];
      u2 = u2 + hash1{ s2[i] - '0', mod2 } *st2[i - best];
    }
    string ans = "1337";
    for (int t = 0; t < 5; t++) {
      ans += char(u.val % mult + 33);
      u.val = u.val / mult;
    }
    for (int t = 0; t < 5; t++) {
      ans += char(u2.val % mult + 33);
      u2.val = u2.val / mult;
    }
    string s3;
    cin >> s3;
    if (ans == s3) cout << "YES";
    else cout << "NO";
  }


}
```

Task D

```
// Online C++ compiler to run C++ program online
#pragma GCC optimize(O3)
#pragma comment(linker, "/stack:2000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
```

```
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <queue>
#include <cmath>
#include <stack>
#include <string>
#include <map>
```

```
#include <unordered_map>
#include <set>
#include <random>
#include <chrono>
#include <deque>
#include <string>
#include <bitset>

using namespace std;

#define int long long

bool skip(int cnt) {
  cout << "1 1\n";
  cout.flush();
  string s;
  cin >> s;
  if (s == "!") return true;
  return false;
}

signed main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  int n;
  cin >> n;
  vector<pair<int, int>> del;
  for (int i = 0; i < 1e9; i++) {
    if (del.size() == 0) {
      cout << "1 1\n";
      cout.flush();
      string s;
      cin >> s;
      if (s == "!") return 0;
      int a2 = (i) % n;
      int b2 = (i + 1) % n;
      if (s == ">" and a2 < b2) {//
        del.push_back({a2, b2  });
      }
      else if (s == "<" and a2 > b2) {
        del.push_back({ a2, b2 });
      }
    }
    else if (del.size() > 0) {
      pair<int, int> u = *del.crbegin();
      del.pop_back();
      cout << (u.first - i % n + n) % n + 1 << " " << (u.second - i % n +
    n) % n + 1 << "\n";
      cout.flush();
      string s;
      cin >> s;
```

```
      if (s == "!") return 0;
    }
  }


}

/*

0 25341
1 53412

1 23415
2 34152
3 41523

3 31524
4 15243
5 52431
6 24315
7 43152

7 23154
8 31542
9 15423
10 54231
11 42315
12 23154
13 31542

13 21543
14 15432
15 54321


*/
```

Task E

```
// Online C++ compiler to run C++ program online
#pragma GCC optimize(O3)
#pragma comment(linker, "/stack:2000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
```

```
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <queue>
#include <cmath>
#include <stack>
#include <string>
```

```
#include <map>
#include <unordered_map>
#include <set>
#include <random>
#include <chrono>
#include <deque>
#include <string>
#include <bitset>

using namespace std;

#define int long long

int n, q;
vector<vector<int>> l(200000);
vector<int> p(200000);
vector<bool> good;
int goodcnt;
vector<int> tin(300000);
vector<int> kek(300000);
int timer = 0;

void dfs(int now, int pred) {
  if (timer > 100100) return;
  tin[timer++] = now;
  p[now] = pred;
  dfs(l[now][(kek[now]++) % l[now].size()], now);
}

vector<int> ans;
int startcnt = 0;

void bfs() {
  queue<int> q;
  queue<int> q2;
  vector<bool> used(n, true);
  q2.push(0);
  used[0] = false;
  while (q2.size() > 0) {
    while (q.size() > 0) {
      int now = q.front();
      q.pop();
      for (int i : l[now]) {
        if (used[i]) {
          used[i] = false;
          if (good[i]) {
            q.push(i);
          }
          else {
            q2.push(i);
          }
        }
      }
```

```
    }
    goodcnt += q2.size();
    ans.push_back(goodcnt * 2);
    startcnt += goodcnt * 2;
    q = q2;
    while (q2.size() > 0) q2.pop();
  }

}


signed main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  cin >> n >> q;
  for (int i = 0; i < n; i++) {
    int k;
    cin >> k;
    for (int j = 0; j < k; j++) {
      int u;
      cin >> u;
      u--;
      l[i].push_back(u);
    }
  }
  dfs(0, -1);
  goodcnt = n;
  good = vector<bool>(n, true);
  for (int i = 0; i < n; i++) {
    if (l[i][0] == p[i]) {
      goodcnt--;
      good[i] = false;
    }
  }
  for (; q > 0; q--) {
    int t;
    cin >> t;
    cout << tin[t] + 1 << "\n";
  }




}

/*
                                                                    ( . .
                                                )
      x
              ,
```

```
            y
      t >= y,
       t <= n,

                ,                                          ,
                                                .                  ,
             ,
               +


*/
```

Task F

```
// Online C++ compiler to run C++ program online
#pragma GCC optimize(O3)
#pragma comment(linker, "/stack:2000")
#pragma GCC optimize("Ofast")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("profile-values,profile-reorder-functions,tracer")
#pragma GCC optimize("vpt")
#pragma GCC optimize("rename-registers")
#pragma GCC optimize("move-loop-invariants")
#pragma GCC optimize("unswitch-loops")
#pragma GCC optimize("function-sections")
#pragma GCC optimize("data-sections")
#pragma GCC optimize("branch-target-load-optimize")
#pragma GCC optimize("branch-target-load-optimize2")
#pragma GCC optimize("btr-bb-exclusive")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
```

```
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define _CRT_SECURE_NO_WARNINGS
#define _USE_MATH_DEFINES
#include <math.h>
#include <stdio.h>
#include <iostream>
#include <algorithm>
#include <fstream>
#include <vector>
#include <queue>
#include <cmath>
#include <stack>
#include <string>
#include <map>
#include <unordered_map>
#include <set>
#include <random>
#include <chrono>
#include <deque>
#include <string>
#include <bitset>

using namespace std;

#define int long long


signed main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout.tie(0);

  int n, m;
  cin >> n >> m;
  vector<string> l(n);
  vector<vector<char>> ans(n, vector<char>(m, '.'));
  int ans2 = 0;
```

```
  for (int i = 0; i < n; i++) cin >> l[i];
  vector<vector<int>> l2(n, vector<int>(m));
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
      l2[i][j] = l[i][j] - '0';
    }
  }
  int startx = 0;
  int starty = 0;
  for (int i = 0; i < n / 2; i++) {
    for (int j = 0; j < m / 2; j++) {
      if (l2[startx + i * 2][starty + j * 2] * l2[startx + i * 2][starty
  + j * 2 + 1] +
          l2[startx + i * 2 + 1][starty + j * 2] * l2[startx + i * 2 + 1][
  starty + j * 2 + 1] >
          l2[startx + i * 2][starty + j * 2] * l2[startx + i * 2 + 1][
  starty + j * 2] +
          l2[startx + i * 2][starty + j * 2 + 1] * l2[startx + i * 2 + 1][
  starty + j * 2 + 1]) {
          ans2 += l2[startx + i * 2][starty + j * 2] * l2[startx + i * 2][
  starty + j * 2 + 1] +
            l2[startx + i * 2 + 1][starty + j * 2] * l2[startx + i * 2 +
  1][starty + j * 2 + 1];
          ans[startx + i * 2][starty + j * 2] = 'L';
          ans[startx + i * 2][starty + j * 2 + 1] = 'R';
          ans[startx + i * 2 + 1][starty + j * 2] = 'L';
          ans[startx + i * 2 + 1][starty + j * 2 + 1] = 'R';
      }
      else {
          ans2 += l2[startx + i * 2][starty + j * 2] * l2[startx + i * 2 +
  1][starty + j * 2] +
            l2[startx + i * 2][starty + j * 2 + 1] * l2[startx + i * 2 +
  1][starty + j * 2 + 1];
          ans[startx + i * 2][starty + j * 2] = 'T';
          ans[startx + i * 2 + 1][starty + j * 2] = 'B';
          ans[startx + i * 2][starty + j * 2 + 1] = 'T';
          ans[startx + i * 2 + 1][starty + j * 2 + 1] = 'B';
      }
    }
  }
  cout << ans2 << "\n";
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
      cout << ans[i][j];
    }
    cout << "\n";
  }


}

/*
```

```
                                                                  ( . .
                                          )
      x
              ,

             y
     t >= y,
      t <= n,

                ,                                          ,
                                                .                  ,
            ,
              +


*/
```