```cpp
#include <stdio.h>
#include <vector>
#include <queue>
#define INF 2000000000000LL
#define pb push_back
using namespace std;

/* Exemplo de uso de dijkstra, resolucao do problema "codeforces 20C - Dijkstra?" */

vector <int> g[100005];
vector <int> w[100005];

int pred[100005];
long long int d[100005];
int seq[100005];

priority_queue < pair <long long int, int> > pq;

int main(void) {
    int n, m;
    int x, y, z;
    int next;
    int viz;

    int i;
    int k;

    scanf(" %d %d", &n, &m);

    for (i = 0; i < m; i++) {
        scanf(" %d %d %d", &x, &y, &z);
        g[x].pb(y);
        w[x].pb(z);
        g[y].pb(x);
        w[y].pb(z);
    }

    for (i = 1; i <= n; i++) {
        d[i] = INF;
    }

    d[1] = 0;
    pred[1] = 1;
    pq.push(make_pair(0, 1));

    while(!pq.empty()) {
        long long int cost = -pq.top().first;
        next = pq.top().second;
        pq.pop();

        if (d[next] < cost) {
            continue;
        }

        for (i = 0; i < (int)g[next].size(); i++) {
            viz = g[next][i];
            if (d[next] + w[next][i] < d[viz]) {
                pred[viz] = next;
                d[viz] = d[next] + w[next][i];
                pq.push(make_pair(-d[viz], viz));
            }
        }
    }

    if (d[n] >= INF) {
        printf("-1\n");
    } else {
        k = 0;
        for (i = n; i != 1; i = pred[i]) {
            seq[k++] = i;
        }
        printf("1");
        for (i = k - 1; i >= 0; i--) {
            printf(" %d", seq[i]);
        }
        printf("\n");
    }

    return 0;
}
```

```cpp
#include <stdio.h>

/* Exemplo de flood fill com DFS, solucao do problema "UVA 572 - Oil Deposits" */

const int MAXN = 105;

char grid[MAXN][MAXN];
int m, n;

int dx[] = {-1, -1, -1, 0, 1, 1, 1, 0};
int dy[] = {-1, 0, 1, 1, 1, 0, -1, -1};

int is_in (int x, int y) {
    return 0 <= x && x < m && 0 <= y && y < n;
}

void DFS(int x, int y) {
    grid[x][y] = '.';

    for (int i = 0; i < 8; i++) {
        int nx = x + dx[i];
        int ny = y + dy[i];

        if (is_in(nx, ny) && grid[nx][ny] == '@') {
            DFS(nx, ny);
        }
    }
    return;
}

int main(void) {

    while(scanf(" %d %d", &m, &n) && m > 0) {
        for (int i = 0; i < m; i++) {
            scanf(" %s", grid[i]);
        }

        int res = 0;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (grid[i][j] == '@') {
                    res++;
                    DFS(i, j);
                }
            }
        }
        printf("%d\n", res);
    }
    return 0;
}

```

```cpp
#include <stdio.h>
#include <vector>
#include <algorithm>
#define pb push_back
#define lli long long int
using namespace std;

/* Exemplo de LCA Em O<NlogN, LogN>, solucao do problema "URI - 1135 - Colônia de Formigas" */

const int MAXN = (int)1e5 + 5;
const int MAX_LOG = 20;

vector <int> g[MAXN];
vector <lli> w[MAXN];

int par[MAXN][MAX_LOG]; // par[x][i] = ancestral de x de distancia 2^i
lli d[MAXN]; // d[x] = distancia da raiz ate o no x
int depth[MAXN]; // depth[x] = profundidade do no x

void DFS(int node, int h, lli dist, int p = -1) {
    par[node][0] = p;
    depth[node] = h;
    d[node] = dist;

    for (int i = 1; i < MAX_LOG; i++) {
        par[node][i] = -1;
        int aux = par[node][i - 1];
        if (aux != -1) {
            par[node][i] = par[aux][i - 1];
        }
    }

    for (int i = 0; i < (int)g[node].size(); i++) {
        int viz = g[node][i];
        if (viz != p) {
            DFS(viz, h + 1, dist + w[node][i], node);
        }
    }
    return;
}

int get_lca(int x, int y) {
    if (depth[x] < depth[y]) {
        swap(x, y);
    }

    for (int i = MAX_LOG - 1; i >= 0; i--) {
        if (par[x][i] != -1 && depth[par[x][i]] >= depth[y]) {
            x = par[x][i];
        }
    }

    if (x == y) {
        return x;
    }

    for (int i = MAX_LOG - 1; i >= 0; i--) {
        if (par[x][i] != par[y][i]) {
            x = par[x][i];
            y = par[y][i];
        }
    }
    return par[x][0];
}

int main(void) {
    int n;
    int a, l;
    int s, t;
    int q;

    while(scanf(" %d", &n) && n) {
        for (int i = 0; i < n; i++) {
            g[i].clear();
            w[i].clear();
        }
        for (int i = 1; i < n; i++) {
            scanf(" %d %d", &a, &l);
            g[a].pb(i);
            w[a].pb(l);
            g[i].pb(a);
            w[i].pb(l);
        }

        DFS(0, 0, 0);
        scanf(" %d", &q);
        for (int i = 0; i < q; i++) {
            if (i != 0) {
                printf(" ");
            }
            scanf(" %d %d", &s, &t);
            int lca = get_lca(s, t);
            lli res = d[s] + d[t] - 2 * d[lca];
            printf("%lld", res);
        }
        printf("\n");
    }
    return 0;
}
```

```
#include <stdio.h>
#define MAXN 105
#define MOD 10000

/* Exemplo de exponenciacao rapida de matrizes para resolucao
        de recorrencias lineares, solucao do problema
        "URI - 1713 - Teletransporte"
*/

int mat[MAXN][MAXN];
int r[MAXN][MAXN];
int aux[MAXN][MAXN];

void multiply(int a[MAXN][MAXN], int b[MAXN][MAXN], int c[MAXN][MAXN], int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            aux[i][j] = 0;
        }
    }
    for (int k = 1; k <= n; k++) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                aux[i][j] += b[i][k] * c[k][j];
                aux[i][j] %= MOD;
            }
        }
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            a[i][j] = aux[i][j];
        }
    }
    return;
}

void fast_exp(int p, int n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            r[i][j] = 0;
        }
        r[i][i] = 1;
    }

    while(p > 0) {
        if (p & 1) {
            multiply(r, r, mat, n);
        }
        multiply(mat, mat, mat, n);
        p /= 2;
    }
    return;
}

int main(void) {
    int n, l;
    int s, t;
    int x;

    while(scanf(" %d %d", &n, &l) != EOF) {
        scanf(" %d %d", &s, &t);
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                mat[i][j] = 0;
            }
            for (int j = 0; j < 4; j++) {
                scanf(" %d", &x);
                mat[i][x]++;
            }
        }
        fast_exp(l, n);
        printf("%d\n", r[s][t]);
    }
    return 0;
}
```

```cpp
#include <stdio.h>
#include <queue>
#define MAXN 50005
#define INF 0x3f3f3f3f
using namespace std;

/* Solucao do problema SUMS da POI  http://main.edu.pl/en/archive/oi/10/sum */

int v[MAXN];
int d[MAXN];

int main(void) {
    int n;
    int k;
    int b;
    priority_queue <pair <int, int> > pq;

    scanf(" %d", &n);
    for (int i = 0; i < n; i++) {
        scanf(" %d", &v[i]);
    }
    d[0] = 0;
    for (int i = 1; i < v[0]; i++) {
        d[i] = INF;
    }
    pq.push(make_pair(0, 0));
    while(!pq.empty()) {
        int cost = pq.top().first;
        int next = pq.top().second;
        pq.pop();

        if (d[next] < cost) {
            continue;
        }
        for (int i = 1; i < n; i++) {
            int viz = (next + v[i]) % v[0];
            if (d[viz] > d[next] + v[i]) {
                d[viz] = d[next] + v[i];
                pq.push(make_pair(-d[viz], viz));
            }
        }
    }
    scanf(" %d", &k);
    while(k--) {
        scanf(" %d", &b);
        if (d[b % v[0]] <= b) {
            printf("TAK\n");
        } else {
            printf("NIE\n");
        }
    }
    return 0;
}
```