T1

先把s左右翻转，然后把6变成9，把9变成6即可
若s含数字2,3,4,5,7中的任何一个，输出NaN

T2

开桶，模拟题意

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const int N = 1e5 + 5;
int n, a[N], b[N], c[N];
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);


    cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    for (int i = 1; i <= n; i++) cin >> b[i];
    for (int i = 1; i <= n; i++) {
        int x; cin >> x;
        c[b[x]]++;
    }

    ll ans = 0;
    for (int i = 1; i <= n; i++) ans += c[a[i]];
    cout << ans;

    return 0;
}
/*
也可以这么写
for (int i = 1; i <= n; i++) cin >> a[i], cnt[a[i]]++;
for (int i = 1; i <= n; i++) cin >> b[i];
for (int i = 1; i <= n; i++) cin >> c[i], ans += cnt[b[c[i]]];
*/
```

T3

bfs/dfs都行 `void dfs(int x, int y, int d)`
把点的坐标和从哪个方向到达该点作为状态 `{x,y,d}`

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const int N = 210;
int n, m, ans, vis[N][N][4];
char g[N][N];
struct { int x, y, d; } q[N * N * 4]; //队列范围=状态数
```

```
int hh, tt = -1;
const int dx[] = {0, 0, 1, -1}, dy[] = {1, -1, 0, 0}; //右左下上
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> (g[i] + 1);

    q[++tt] = {2, 1, 0};

    while (hh <= tt) {
        int x = q[hh].x, y = q[hh].y, d = q[hh].d;
        ++hh;
        int xx = x + dx[d], yy = y + dy[d];
        if (g[xx][yy] == '#') {
            for (int i = 0; i < 4; i++)
                if (!vis[x][y][i]) vis[x][y][i] = true, q[++tt] = {x, y, i};
        } else if (!vis[xx][yy][d]) {
            if (g[xx][yy] == '.') ++ans, g[xx][yy] = 'o';
            vis[xx][yy][d] = true, q[++tt] = {xx, yy, d};
        }
    }
    cout << ans;

    return 0;
}
```

T4

题意：询问 u的子树中深度为dis的结点数量

做法1：离线处理询问

`cnt[x]` 数组记录深度为x的结点数。

dfs访问u的子树前后，cnt[dis]的变化量就是u的子树中深度为dis的结点数量，也就是答案。

思路与洛谷P3605类似（但不用树状数组），可以看一下

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const int N = 2e5 + 5;
vector<int> g[N];
vector<pair<int, int>> qry[N]; //D,id
int n, q, cnt[N], ans[N];

void dfs(int u, int dep = 0) {
    for (auto p : qry[u]) ans[p.second] -= cnt[p.first];
    ++cnt[dep];
    for (int v : g[u]) dfs(v, dep + 1);
    for (auto p : qry[u]) ans[p.second] += cnt[p.first];
}

int main() {
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n;
    for (int i = 2, p; i <= n; i++)
        cin >> p, g[p].push_back(i);
    cin >> q;
    for (int i = 1; i <= q; i++) {
        int u, d;
        cin >> u >> d;
        qry[u].push_back({d, i}); //离线
    }
    dfs(1);
    for (int i = 1; i <= q; i++)
        cout << ans[i] << '\n';

    return 0;
}
```

做法2：dfs序/时间戳

做法详见原题abc202e题解。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 2e5 + 5;
//时间timer 1 2 3 4 5 6 7 8 9 10 11 12 13 14
//结点      1 2 2 3 4 6 6 4 5  5  7  7  3  1
int n, q, in[N], out[N], timer;
vector<int> e[N], d[N];
void dfs(int u, int dep) {
    in[u] = ++timer;
    d[dep].push_back(timer);
    for (int v : e[u]) dfs(v, dep + 1);
    out[u] = ++timer;
}
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    cin >> n;
    for (int i = 2, x; i <= n; ++i) {
        cin >> x;
        e[x].push_back(i);
    }
    dfs(1, 0);
    cin >> q;
    while (q--) {
        int u, dis;
        cin >> u >> dis;
        cout << lower_bound(d[dis].begin(), d[dis].end(), out[u]) -
lower_bound(d[dis].begin(), d[dis].end(), in[u]) << '\n';
    }
    return 0;
}
```