// Program to demonstrate tree traversal with

// ability to jump between nodes of same height

#include <bits/stdc++.h>

using namespace std;

#define N 1000

vector<int> H[N];

// Arrays declaration

int Fruit[N];

int Parent[N];

int dp[N][20];

// Function for DFS

void dfs1(vector<int> tree[], int s,

          int p, int h)

{

    Parent[s] = p;

    int i;

    H[h].push\_back(s);

    for (i = 0; i < tree[s].size(); i++) {

        int v = tree[s][i];

        if (v != p)

            dfs1(tree, v, s, h + 1);

    }

}

// Function for DFS

int dfs2(vector<int> tree[], int s,

         int p, int h, int j)

{

    int i;

    int ans = 0;

    if (dp[s][j] != -1)

        return dp[s][j];

    // jump

    if (j > 0) {

        for (i = 0; i < H[h].size(); i++) {

            int v = H[h][i];

            if (v != s)

                ans = max(ans, dfs2(tree, v,

                        Parent[v], h, j - 1));

        }

    }

    // climb

    for (i = 0; i < tree[s].size(); i++) {

        int v = tree[s][i];

        if (v != p)

            ans = max(ans, dfs2(tree, v, s, h + 1, j));

    }

    if (Fruit[s] == 1)

        ans++;

    dp[s][j] = ans;

    return ans;

}

int maxFruit(vector<int> tree[],

             int NodesWithFruits[],

             int n, int m, int k)

{

    // reseting dp table and Fruit array

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < 20; j++)

            dp[i][j] = -1;

        Fruit[i] = 0;

    }

    for (int i = 0; i < m; i++)

        Fruit[NodesWithFruits[i]] = 1;

    dfs1(tree, 1, 0, 0);

    int ans = dfs2(tree, 1, 0, 0, k);

    return ans;

}

void addEdge(vector<int> tree[], int u, int v)

{

    tree[u].push\_back(v);

    tree[v].push\_back(u);

}

int main()

{

    int n = 12; // Number of nodes

    int k = 2; // Number of allowed jumps

    vector<int> tree[N];

    // Edges

    addEdge(tree, 1, 2);

    addEdge(tree, 1, 3);

    addEdge(tree, 2, 4);

    addEdge(tree, 2, 5);

    addEdge(tree, 5, 9);

    addEdge(tree, 9, 10);

    addEdge(tree, 9, 11);

    addEdge(tree, 11, 12);

    addEdge(tree, 1, 3);

    addEdge(tree, 3, 7);

    addEdge(tree, 7, 6);

    addEdge(tree, 7, 8);

    int NodesWithFruits[] = { 2, 4, 5, 7, 8, 9, 11, 12 };

    // Number of nodes with fruits

    int m = sizeof(NodesWithFruits) / sizeof(NodesWithFruits[0]);

    int ans = maxFruit(tree, NodesWithFruits, n, m, k);

    cout << ans << endl;

    return 0;

}