1.(奇数区间)给定一个长度为 n 的数列：a1,a2,⋯an，如果其中一段连续的子序列 ai,ai+1,⋯aj(i≤j)中，奇数比偶数多，我们就称这个区间[i,j] 是奇数区间。求给定的n个数中有多少个奇数区间。

#include<bits/stdc++.h>

using namespace std;

const int off = 1e6 + 1;

const int maxn = 1e6 + 5;

int a[maxn], s[maxn], c[2 \* maxn];

int n;

long long ans;

int lowbit(int x){

return x & -x;

}

long long getSum(int x){

long long res = 0;

while(x > 0){

res += c[x];

① ;

}

return res;

}

void add(int x, int k){

while( ② ){

c[x] += k;

③ ;

}

}

int main(){

cin >> n;

add(off, 1);

for(int i = 1; i <= n; i++){

cin >> a[i];

s[i] = ④ ;

ans += getSum( ⑤ );

add(s[i] + off,1);

}

cout << ans << endl;

return 0;

}

（1）①处应该填（ A ）

A. x -= lowbit(x)

B. x = lowbit(x)

C. x += lowbit(x)

D. x--

（2）②处应该填（ D ）

A. x < off

B. x <= n

C. x != 0

D. x <= 2 \* off

（3）③处应该填（ C ）

A. x -= lowbit(x)

B. x = lowbit(x)

C. x += lowbit(x)

D. x++

（4）④处应该填（ C ）

A. s[i - 1] + a[i] % 2 ? 1 : -1

B. s[i - 1] + a[i] % 2 == 1 ? -1 : 1

C. s[i - 1] + a[i] % 2 != 0 ? 1 : -1

D. s[i - 1] + !a[i] % 2 ? 1 : -1

（5）⑤处应该填（ B ）

A. 0

B. s[i] + off - 1

C. s[i - 1]

D. s[i] + off

2.（最近公共祖先）已知一棵有根的多叉树，每次询问下求给定的两个点的最近公共祖先。其中输入的第一行三个整数为结点个数，询问次数，根节点序号。

最近公共祖先：在一棵有根树中，一个结点的祖先结点是指它本身或者它父节点的祖先。对于任意两个结点的公共祖先中距离两者最近的结点。

#include <bits/stdc++.h>

using namespace std;

const int MAXN = 500010;

vector<int> edge[MAXN];

int depth[MAXN], lg[MAXN];

int pa[MAXN][22];

int n, m, r, x, y;

void dfs(int fa, int sn, int dep) {

depth[sn] = dep;

pa[sn][0] = fa;

for(int i = 1; ① ; i++) {

pa[sn][i] = pa[pa[sn][i - 1]][i - 1];

}

for(int i = 0; i < edge[sn].size(); i++) {

int fn = edge[sn][i];

if (fn == fa) {

continue;

}

dfs( ② );

}

}

int solve(int x, int y) {

if (depth[x] < depth[y]) {

swap(x, y);

}

while(depth[x] > depth[y]) {

x = ③ ;

}

if(x == y) {

return x;

}

for(int k = lg[depth[x]] - 1; k >= 0; --k) {

if( ④ ) {

x = pa[x][k], y = pa[y][k];

}

}

return pa[x][0];

}

int main() {

cin >> n >> m >> r;

for(int i = 1; i <= n; i++)

lg[i] = lg[i - 1] + (1 << lg[i - 1] == i);

for (int i = 0; i < n - 1; i++) {

cin >> x >> y;

edge[x].push\_back(y);

edge[y].push\_back(x);

}

dfs( ⑤ );

while (m--) {

cin >> x >> y;

cout << solve(x, y) << endl;

}

return 0;

}

（1）①处应该填（ D ）

A. i < pa[sn].size();

B. i <= lg[depth[fn]

C. i < lg[dep]

D. i <= lg[depth[sn]]

（2）②处应该填（ A ）

A. sn, fn, dep + 1

B. fn, sn, dep + 1

C. sn, fn, dep

D. fn, sn, dep

（3）③处应该填（ B ）

A. lg[depth[x] - depth[y]]

B. pa[x][lg[depth[x] - depth[y]] - 1]

C. pa[y][lg[depth[x]

D. pa[y][lg[depth[y] - depth[x]]

（4）④处应该填（ C ）

A. pa[k][y] == pa[k][x]

B. !pa[x][k]

C. pa[x][k] != pa[y][k]

D. pa[x][k] == pa[y][k]

（5）⑤处应该填（ B ）

A. r, 0, 0

B. 0, r, 0

C. r, 0, 1

D. 0, r, 1

1.第一题考察的点是树形数组，需要对树形数组的建立，修改，查找的知识点有所了解。①可以直接通过while循环的循环条件就能判断出来。②需要了解由于该程序是采用前缀和的形式，为奇数+1，偶数-1的写法，所以可能一个区间的和会产生负数，但是没有负数下标，所以为了把下标统一为正数，结果需要加上一个off，那么根据主函数内需要对off区间全体置为1，所以需要小于2倍的off防止越界；③由while循环的循环条件及对树形数组的知识了解可以直接得出答案。④考虑到求区间奇数比偶数的个数多多少个需要用到前缀和，④即考察基本前缀和的1,-1形式。⑤理解题，需要根据对这道题目的理解，判断出正确的公式，并将结果更新到树形数组中。

2。第二题考察的是最近公共祖先的倍增写法。需要掌握最近公共祖先的基本解法以及对倍增原理有所了解。①考察阅读程序的仔细程度，以及对倍增数组理解透彻，这个循环是初始化倍增父节点。所以循环的终止条件为当前树的深度的倍增数组的值。②考察DFS函数的阅读理解能力，只需要对DFS的三个传参有基本的了解以及懂得三个变量的作用就可以直接得出答案。③该点考察倍增优化的LCA核心算法，根据当前结果更新最近父节点的过程。④该点只需对前后文阅读理解通透，明白pa数组的第一维表示结点编号，第二维表示倍增的数，即可填出答案。⑤送分题，主函数内dfs的传参。