```cpp
/*
    all nodes are completely renamed
    BC contains the block-cut tree
    ord is total nodes in BCT
    u in original graph is in node[u] in BCT
    if ap[u] == true, then u is articulation point in original graph
    if fake[u] == true, then u is a fake node in BCT, else this represents a
biconnected component
*/

ll dis[MAX], low[MAX], par[MAX], ap[MAX], ord;
ll node[MAX], fake[MAX];
vl G[MAX], BC[MAX], tmp;
stack<pl>stk;

void dfs_bc(ll u = -1, ll v = -1)
{
    ll id = ++ord;
    tmp.clear();
    while (!stk.empty()){
        pl t = stk.top();
        stk.pop();
        tmp.pb(t.x), tmp.pb(t.y);
        if (t == mp(u, v)) break;
    }

    sort(tmp.begin(), tmp.end());
    tmp.erase(unique(tmp.begin(), tmp.end()), tmp.end());

    for (ll i=0; i<tmp.size(); i++){
        ll u = tmp[i];
        if (ap[u]){
            if (node[u] == -1) node[u] = ++ord;
            fake[node[u]] = true;

            BC[node[u]].pb(id);
            BC[id].pb(node[u]);
        }
        else node[u] = id;
    }
}

void dfs_vis(ll u)
{
    static ll time = 0;
    ll cld = 0;

    dis[u] = low[u] = ++time;

    for (ll i=0; i<G[u].size(); i++){
        ll v = G[u][i];
        if (dis[v] == -1){
            stk.push(mp(u, v));
            cld++;
            par[v] = u;
            dfs_vis(v);

            low[u]  = min(low[u], low[v]);
            if ((par[u] == -1 && cld > 1) || (par[u] != -1 && low[v] >= dis[u])){
                ap[u] = true;
                dfs_bc(u, v);
            }
        }
        else if (v != par[u] && dis[v] < dis[u])
        {
            stk.push(mp(u, v));
```

```
            low[u]  = min(low[u], dis[v]);
        }
    }

    if (par[u] == -1) dfs_bc();
}

void block_cut(ll n)
{
    ord = 0;
    mem(dis, -1);
    mem(ap, false);
    mem(par, -1);
    mem(node, -1);
    mem(fake, false);

    for (ll i=1; i<=n; i++) if (dis[i] == -1) dfs_vis(i);
}
```