```
class Solution {
    public List<Integer> killProcess(List<Integer> pid, List<Integer> ppid, int kill) {
        List<Integer> res = new ArrayList<>();
        Map<Integer, List<Integer>> mp = new HashMap<>();
        for (int i = 0; i < pid.size(); i++) {
            int p = ppid.get(i);
            int c = pid.get(i);
            if (!mp.containsKey(p)) {
                mp.put(p, new ArrayList<>());
            }
            mp.get(p).add(c);
        }

        Queue<Integer> q = new LinkedList<>();
        q.offer(kill);
        while (!q.isEmpty()) {
            int id = q.poll();
            res.add(id);
            if (mp.containsKey(id)) {
                for (int i : mp.get(id)) {
                    q.offer(i);
                }
            }
        }
        return res;
    }
}
```

Given **n** processes, each process has a unique **PID (process id)** and its **PPID (parent process id)**.

Each process only has one parent process, but may have one or more children processes. This is just like a tree structure. Only one process has PPID that is 0, which means this process has no parent process. All the PIDs will be distinct positive integers.

We use two list of integers to represent a list of processes, where the first list contains PID for each process and the second list contains the corresponding PPID.