B Tree

Given the structure of a tree, detect whether it can be a valid B tree. Consider the following tree description.

```
5
6
6 0 1 2
2 6 2 3
4 2 3 0
5 2 4 0
1 2 5 0
3 6 3 0
```

The first line describes the number of keys that can be stored in each node. This is a tree where each node can contain at the most five keys.

The second line describes the number of nodes in the tree. This tree has six nodes. If the tree has N nodes, then assume that they will be numbered from 1 to N.

Next each line describes a node. Each line four fields: Node id, Parent id, Number of keys in the node, Number of children of the node. The root node does not have any parent and it is indicated by value 0 in the corresponding field. The nodes can appear in the file in any order with only one guarantee that a parent is always described before its children. Nodes that have zero children, are the leaf nodes.

The third line describes the root node with node id 6 and parent id 0. This node has only one key and two children. The fourth line describes the node with id 2 and parent id 6. This node contains two keys and three children.

Given such a structure, your code should check if this tree can be a valid B tree.

If the answer is yes, simply output the message “This can be a valid B Tree”.

If the answer is no, then provide one of the following reasons.

1. <Node id> does not have minimum occupancy.

2. <Node id> does not satisfy the requirement that Children count = Key count + 1.

3. <Node id1> and <Node id2> are a pair of leaf nodes that do not have the same depth.

For some input trees, multiple reasons can be correct. In some cases, only one reason can be correct with multiple possible node ids. Providing just one instance of B tree requirement violation is enough is enough. Assume that any input given will not have errors beyond the mentioned three possible types of errors.

How your code should compile: g++ main.cpp

How your code should run: ./a.out <input1.txt >output1.txt