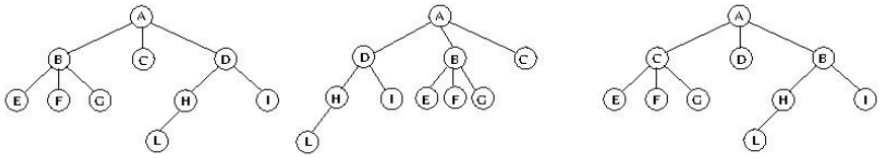
A n-ary tree is defined using the data structure reported below. Nodes have a string as a key, and a number of children equal to n_child. The array child individuates these children.

```
typedef struct node_s node_t;
struct node_s {
  char *key;
  int n_child;
  node_t **child;
};
```

For each tree, there is a corresponding set of paths, which is the set of all paths starting from the root and reaching all leaves.
Two trees are defined to be equivalent if and only if they generate exactly the same set of paths.

For example, the leftmost tree and the one in the middle are equivalent but the rightmost is not. The two trees on the left-hand side of the picture generate the following set of paths: {ABE, ABF, ABG, AC, ADHL, ADI}. On the contrary, the rightmost tree generates the set of paths: {ACE, ACF, ACG, AD, ABHL, ABI}. Notice that, in this example, all keys are single characters only for the sake of mutual understanding. More generally, keys are strings of unknown length.



Write the function

```
int equivalent (node_t *root1, node_t *root2);
```

to verify whether two trees are equivalent or not. The function must return 1 if the two trees are equivalent (they do generate the same set of paths) and must return 0, otherwise.