Lab 2

YAO ZHAO

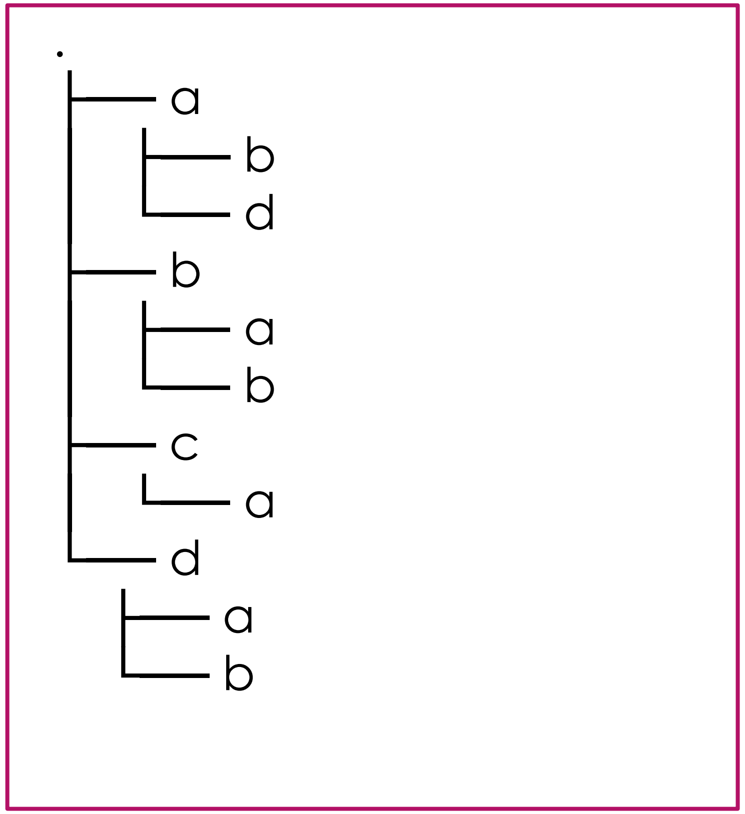
Lab2.A

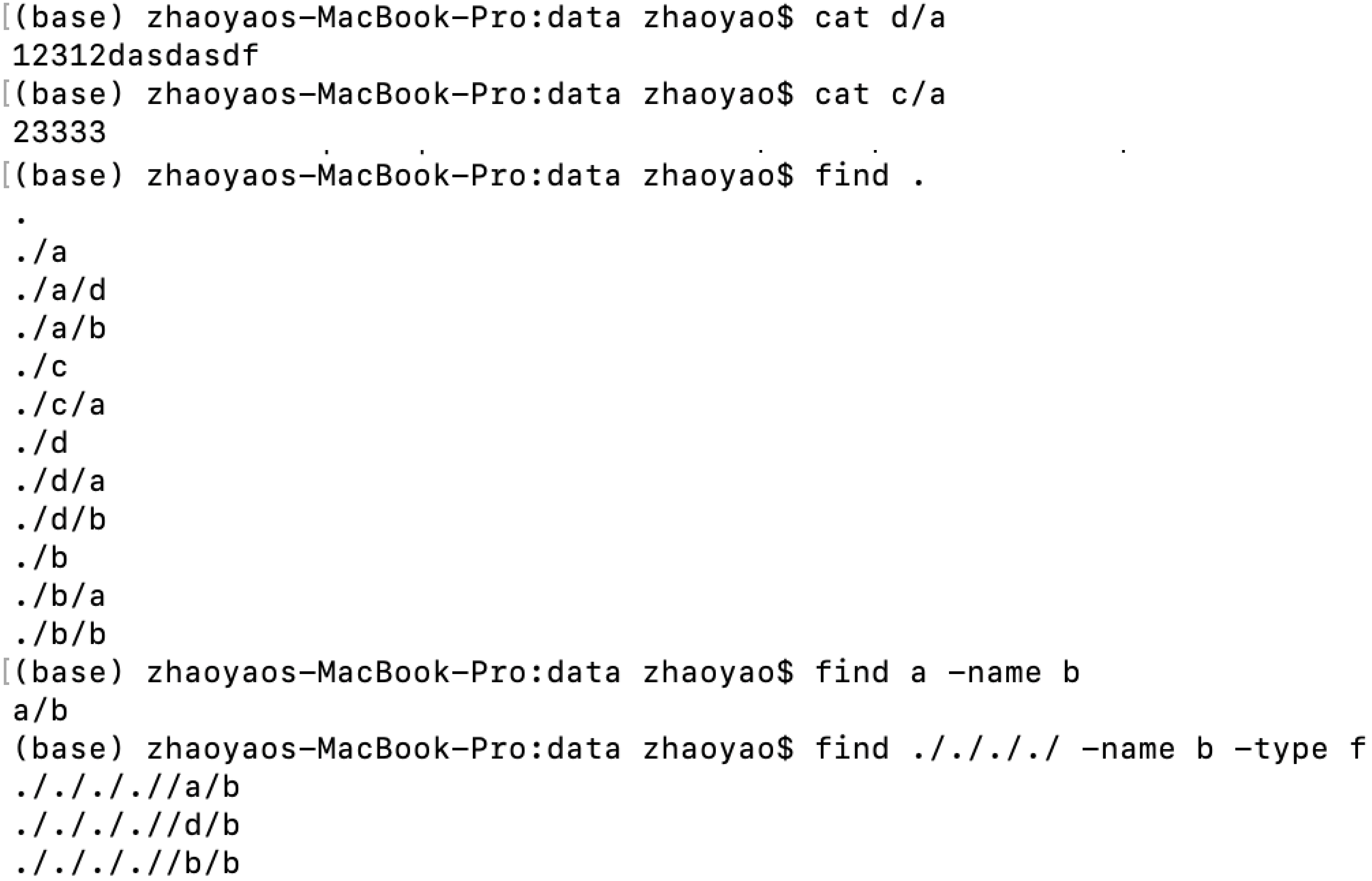
https://spaces.sustech.cloud/classes/14/assignment/lab2

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

Tree:





```
[(base) zhaoyaos-MacBook-Pro:data zhaoyao$ cat d/a
12312dasdasdf
[(base) zhaoyaos-MacBook-Pro:data zhaoyao$ cat c/a
23333
[(base) zhaoyaos-MacBook-Pro:data zhaoyao$ find .
.
./a
./a/d
./a/b
./c
./c/a
./d
./d/a
./d/b
./b
./b/a
./b/b
[(base) zhaoyaos-MacBook-Pro:data zhaoyao$ find a -name b
a/b
(base) zhaoyaos-MacBook-Pro:data zhaoyao$ find ./././ -name b -type f
.//././/a/b
.//././/d/b
.//././/b/b
```

This problem only requires building and traversing a file tree, which can be done using Depth-First Search (DFS).

How to building a tree: to build a tree data structure, you can create a class to represent the nodes of the tree and then establish the parent-child relationships between the nodes.

Tree Node:

fields	name	File or file folder’s name
	type	The node’s type is file or file folder
	content	If The node is file, store the content here, if not a file, the content can be null
	Father node	Set to null if the node is root
	Children list	Set to null or set the size is 0 if The node is file

Initial state:

root

name: .

type: folder

content: null

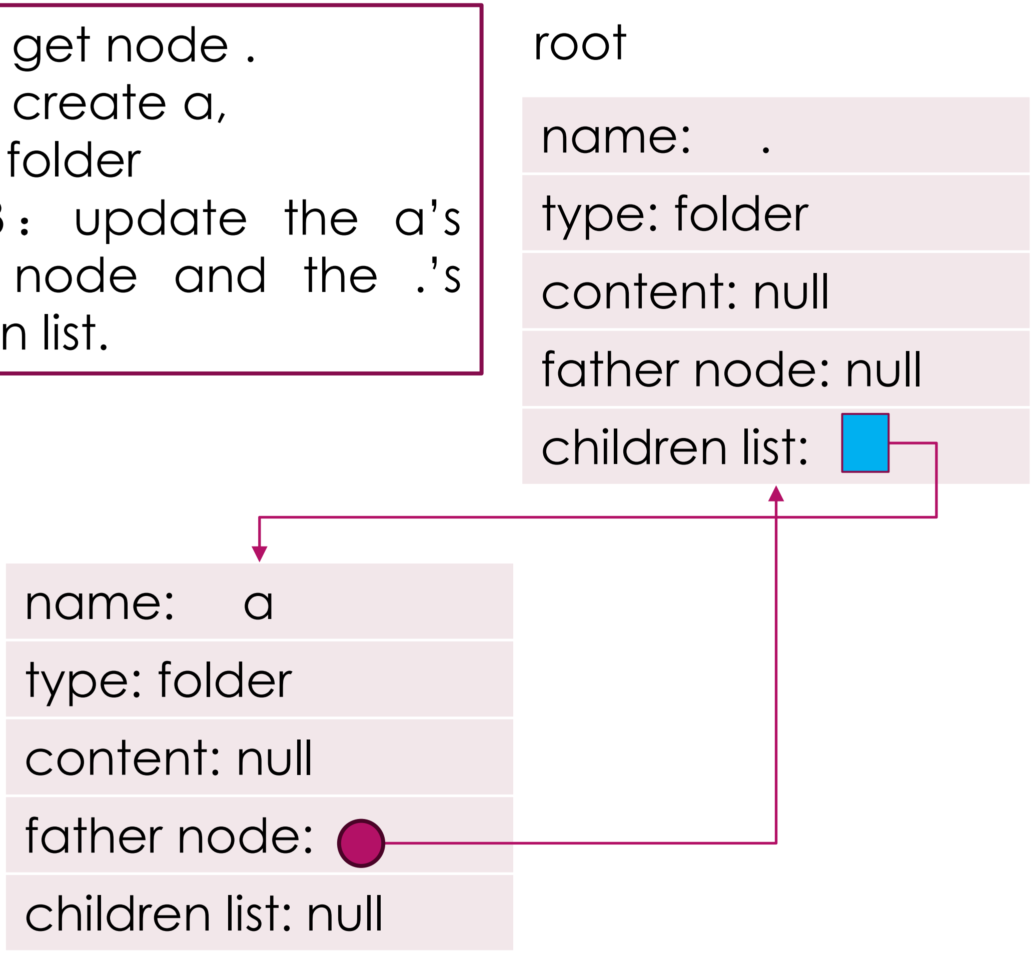
father node: null

children list: null

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

Step 1: get node .
Step 2: create a,
Type is folder
Step 3: update the a’s father node and the .’s children list.



root

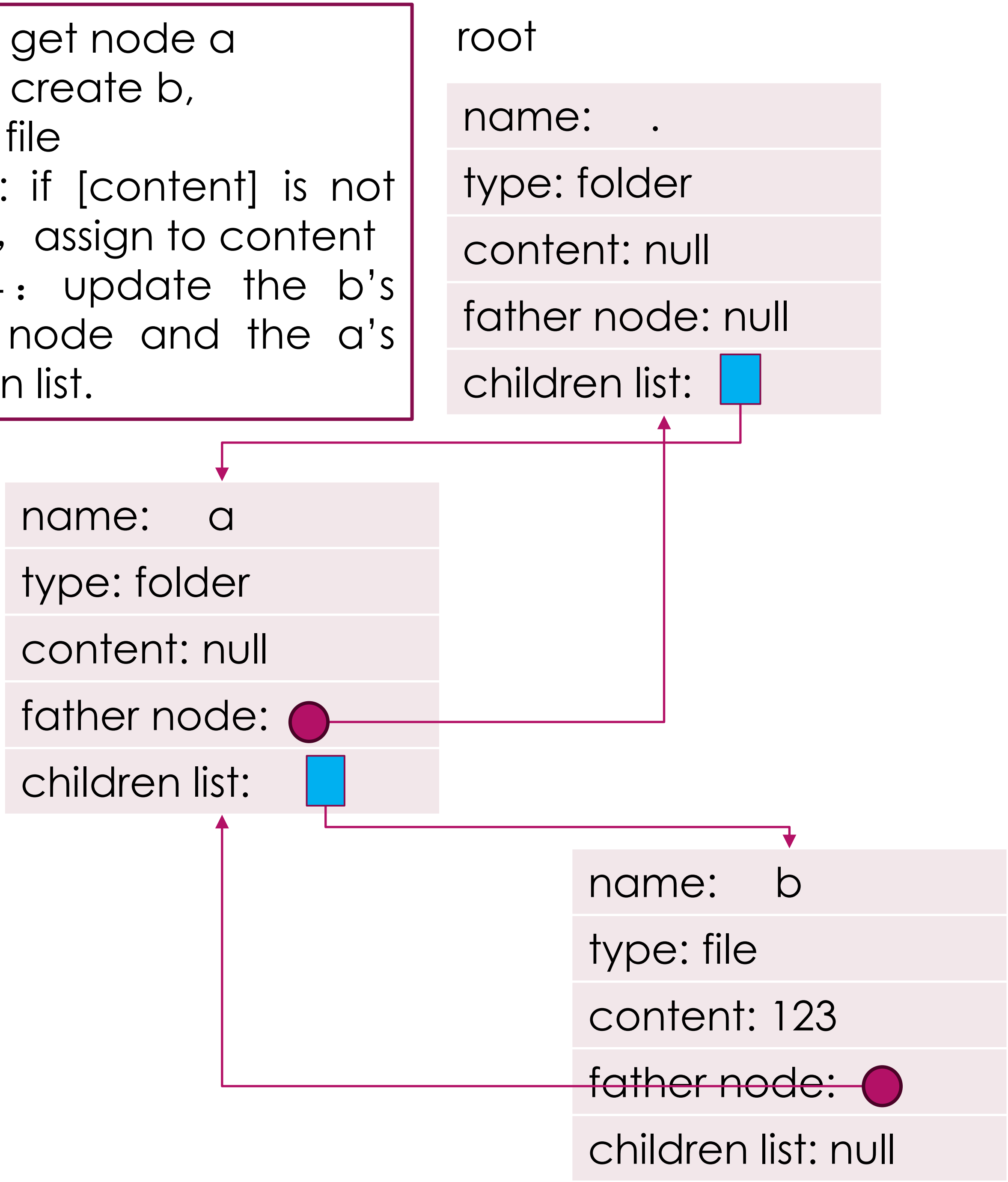
name: .
type: folder
content: null
father node: null
children list: 🟦

name: a
type: folder
content: null
father node: ●
children list: null

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

Step 1: get node a
Step 2: create b,
Type is file
Step 3: if [content] is not empty，assign to content
Step 4：update the b's father node and the a's children list.



root

name: .
type: folder
content: null
father node: null
children list:

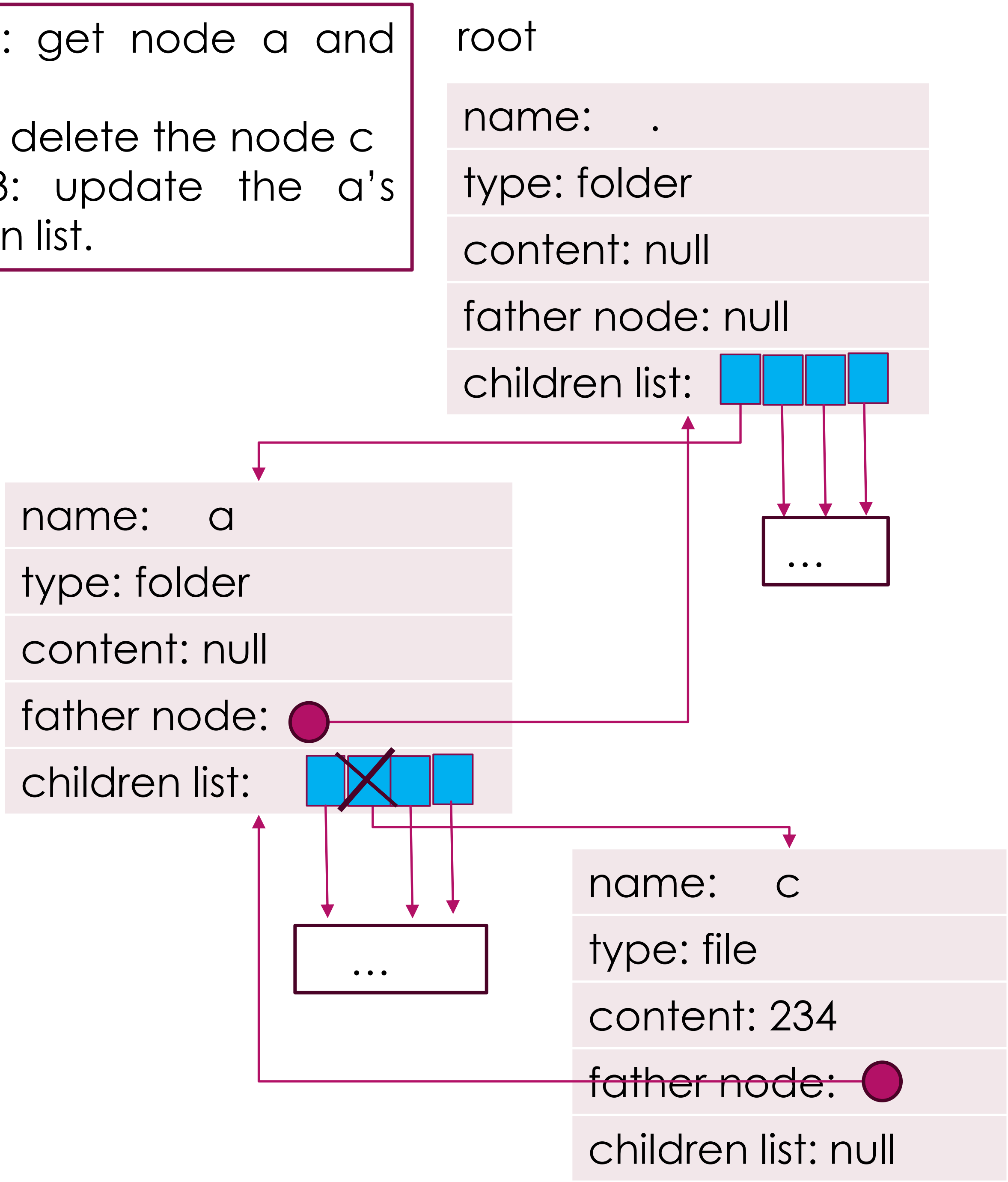
name: a
type: folder
content: null
father node:
children list:

name: b
type: file
content: 123
father node:
children list: null

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

Step 1: get node a and a/c
Step 2: delete the node c
Step 3: update the a's children list.



root

name: .
type: folder
content: null
father node: null
children list:

...

name: a
type: folder
content: null
father node: ●
children list:

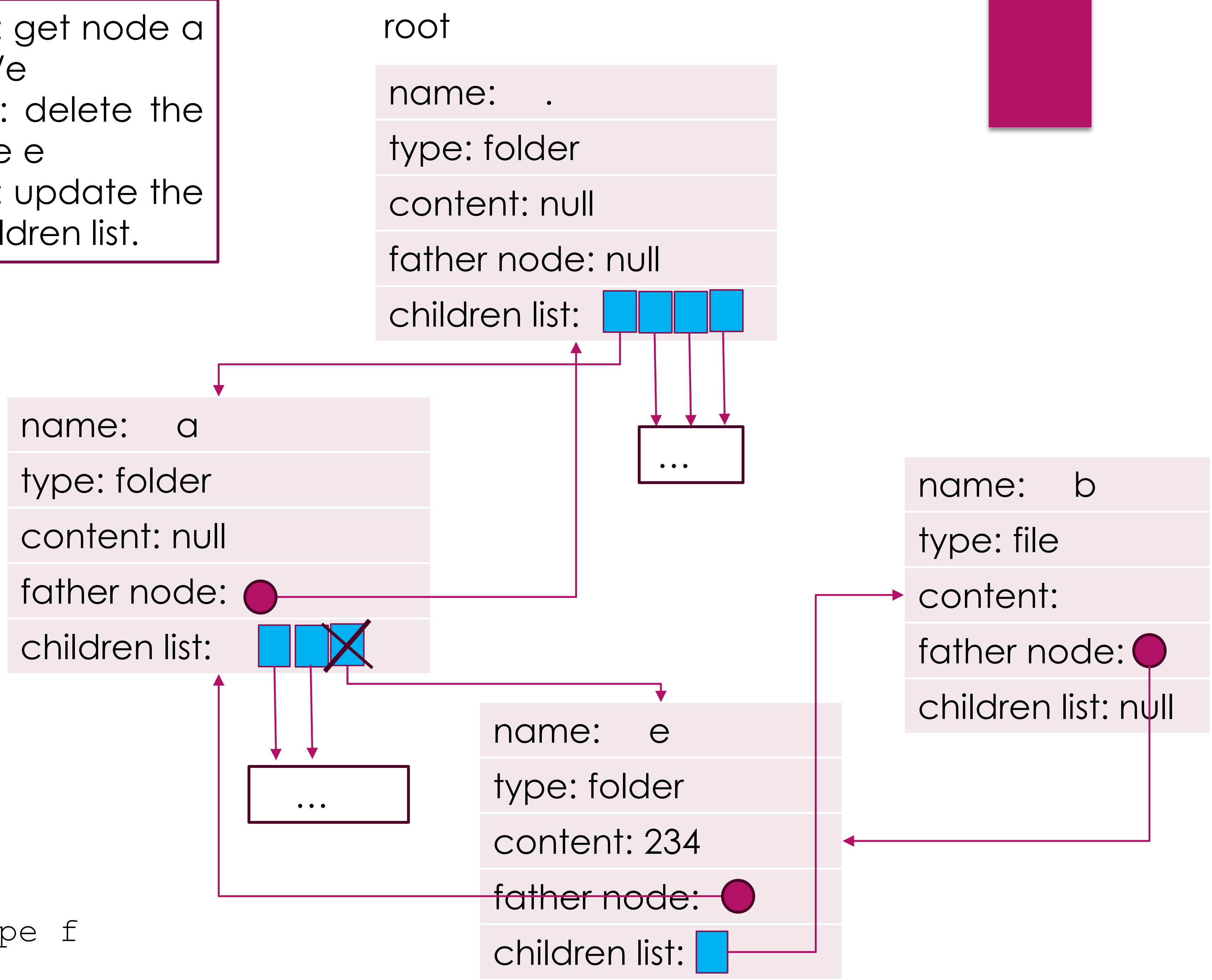
...

name: c
type: file
content: 234
father node: ●
children list: null

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

Step 1: get node a and a/e
Step 2: delete the subtree e
Step 3: update the a's children list.



root

name: .
type: folder
content: null
father node: null
children list:

name: a
type: folder
content: null
father node:
children list:

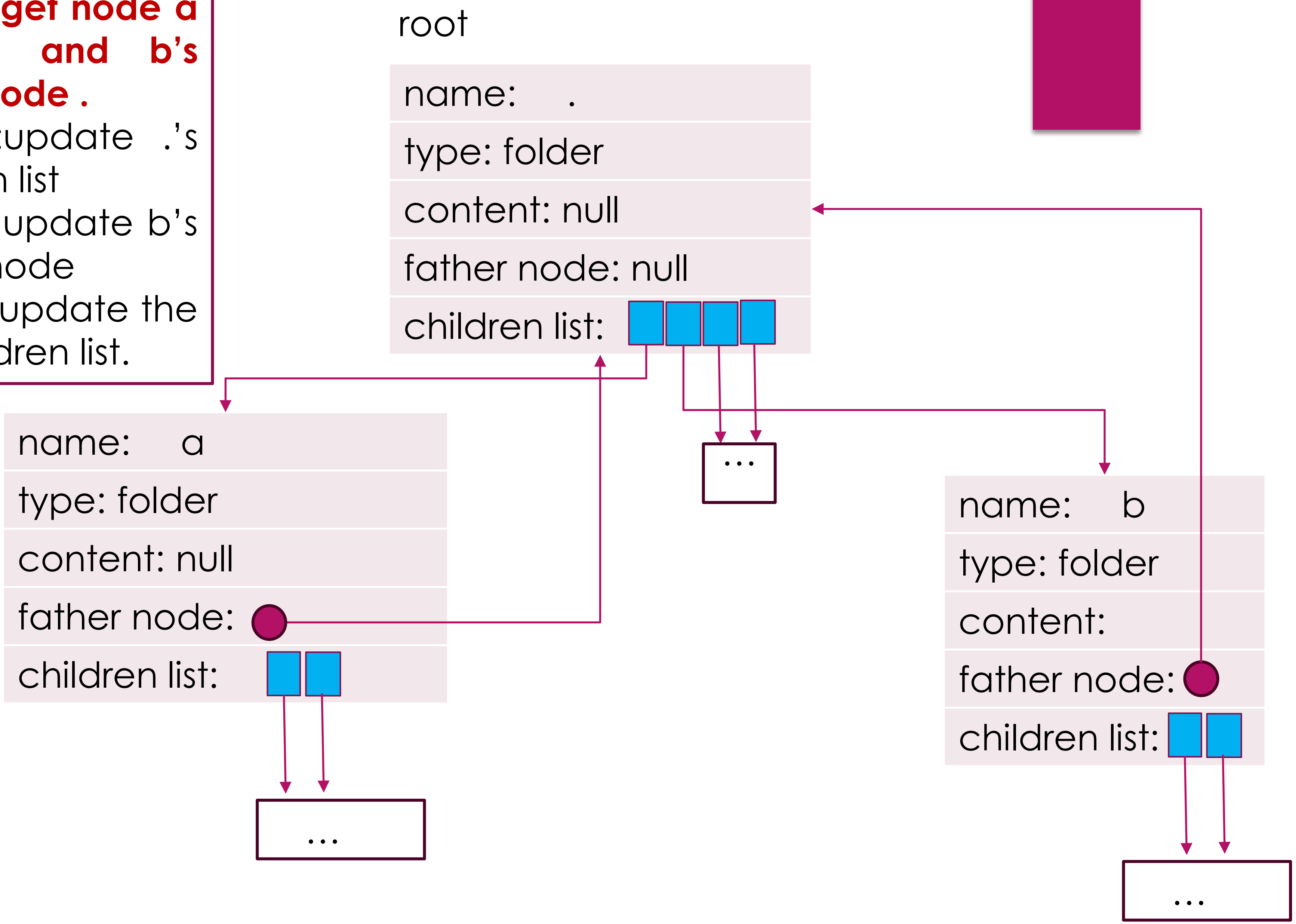
name: b
type: file
content:
father node:
children list: null

name: e
type: folder
content: 234
father node:
children list:

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
mv b a
cat d/a
cat c/a
find
find a -name b
find ././././ -name b -type f
```

Step 1: get node a and b and b's father node .
Step 2:update .'s children list
Step 3: update b's father node
Step 3: update the a's children list.



root

name: .
type: folder
content: null
father node: null
children list:

name: a
type: folder
content: null
father node: ●
children list:

name: b
type: folder
content:
father node: ●
children list:

Input:

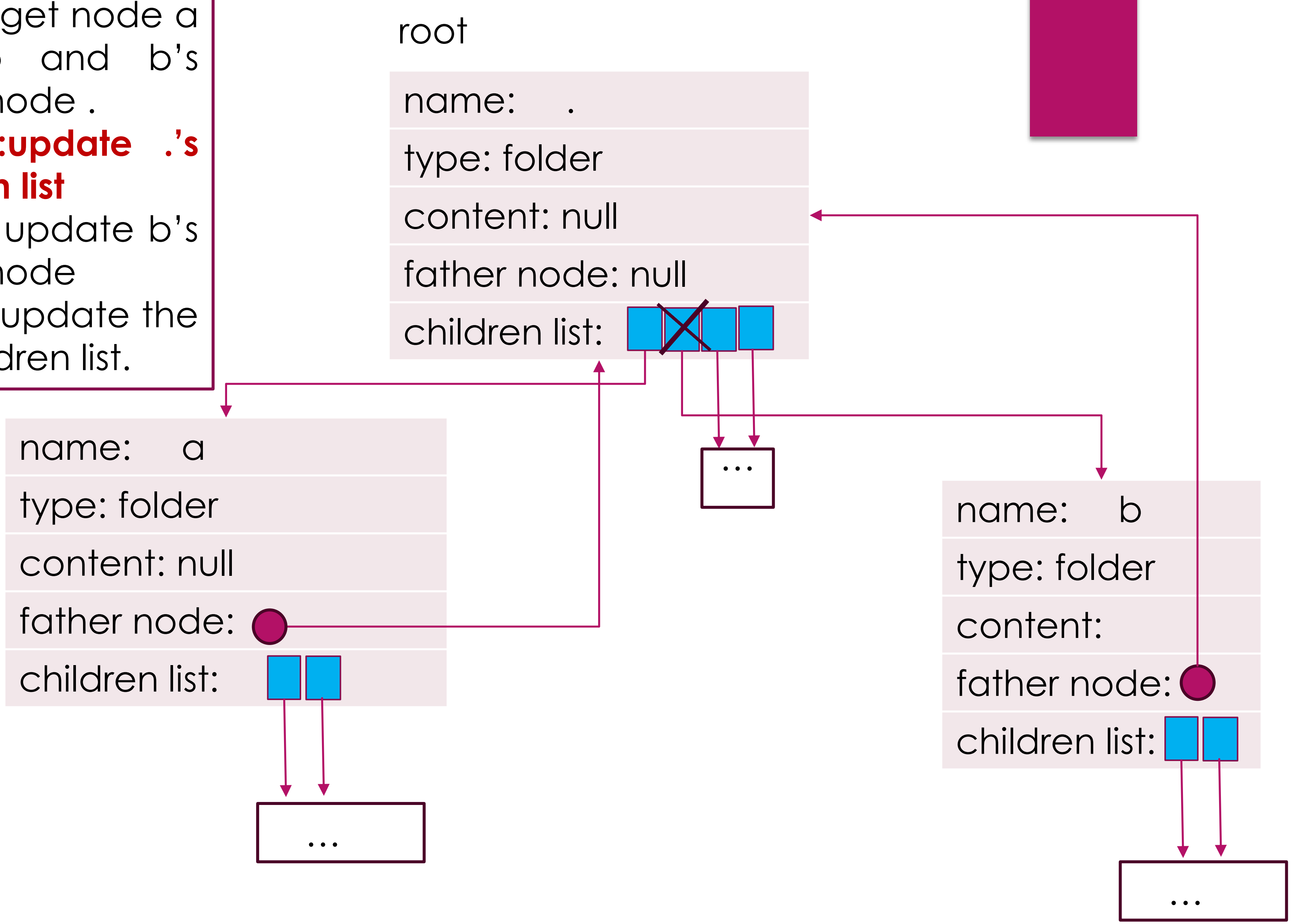
```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
mv b a
cat d/a
cat c/a
find
find a -name b
find ././././ -name b -type f
```

Step 1: get node a and b and b's father node .

Step 2:update .'s children list

Step 3: update b's father node

Step 3: update the a's children list.



root

name: .

type: folder

content: null

father node: null

children list:

name: a

type: folder

content: null

father node: ●

children list:

…

…

name: b

type: folder

content:

father node: ●

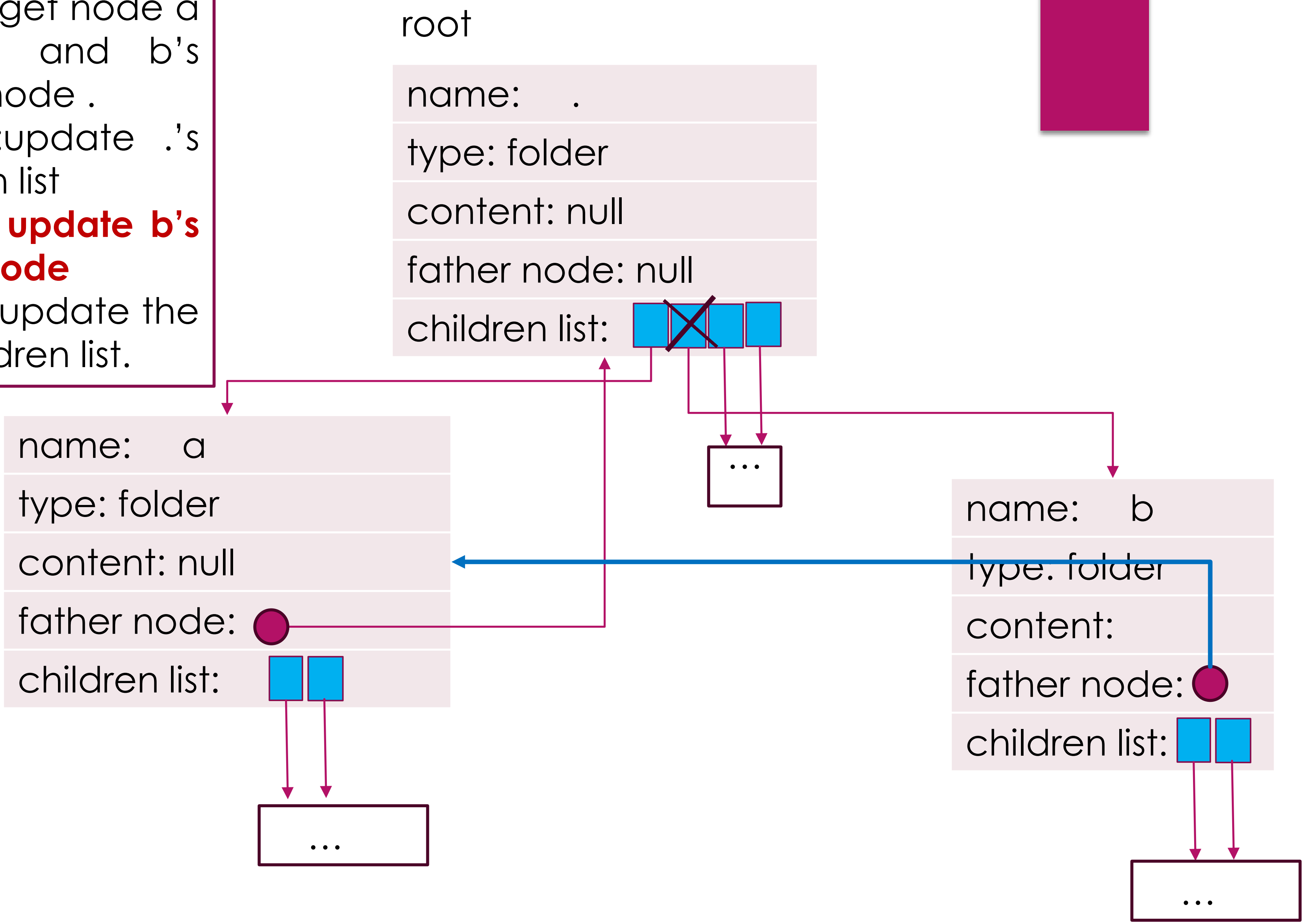
children list:

…

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
mv b a
cat d/a
cat c/a
find
find a -name b
find ././././ -name b -type f
```

Step 1: get node a and b and b's father node .
Step 2:update .'s children list
Step 3: update b's father node
Step 3: update the a's children list.



root

name: .
type: folder
content: null
father node: null
children list:

...

name: a
type: folder
content: null
father node:
children list:

...

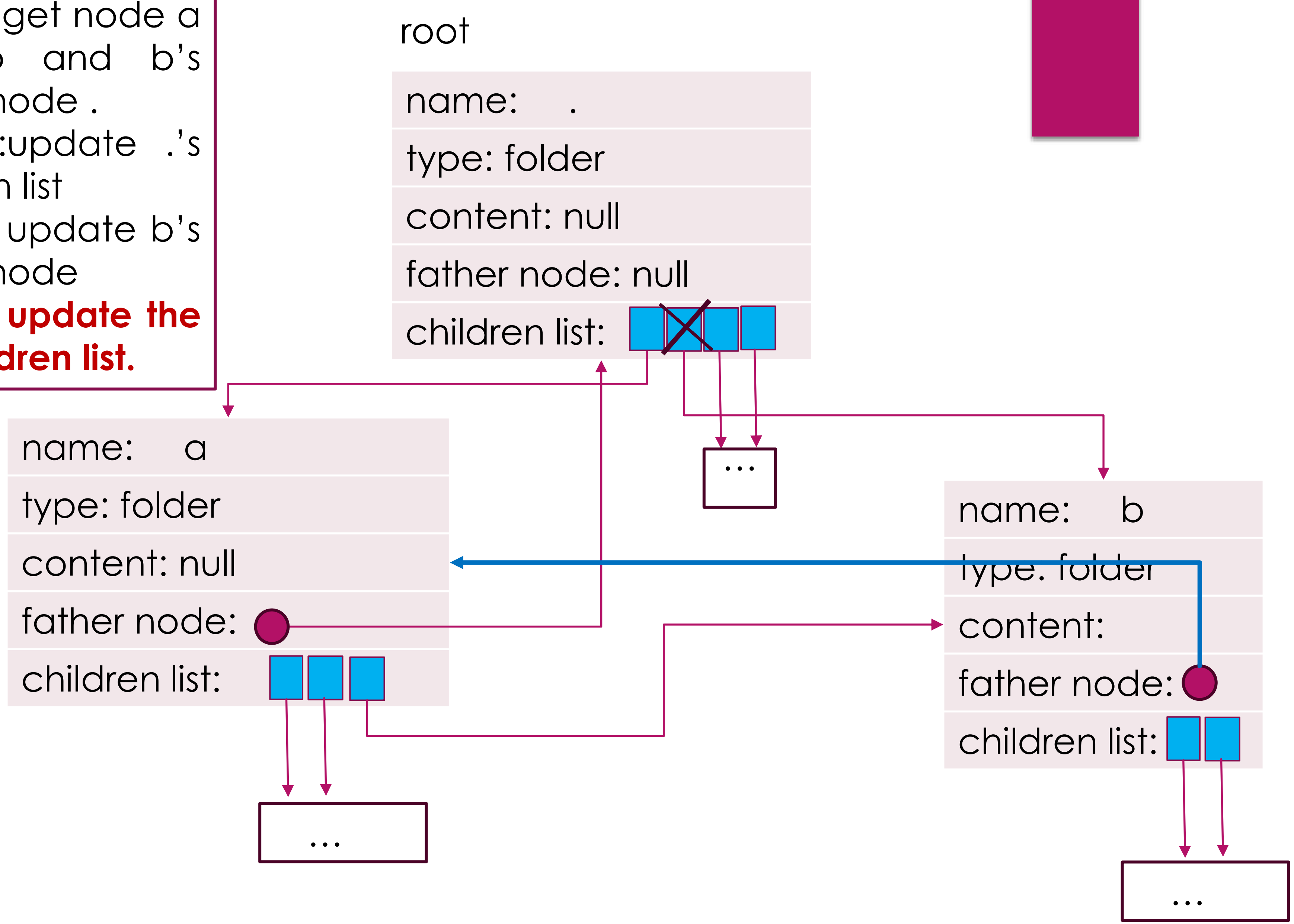
name: b
type: folder
content:
father node:
children list:

...

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
mv b a
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

Step 1: get node a and b and b's father node .
Step 2:update .'s children list
Step 3: update b's father node
Step 3: update the a's children list.



root

name: .
type: folder
content: null
father node: null
children list:

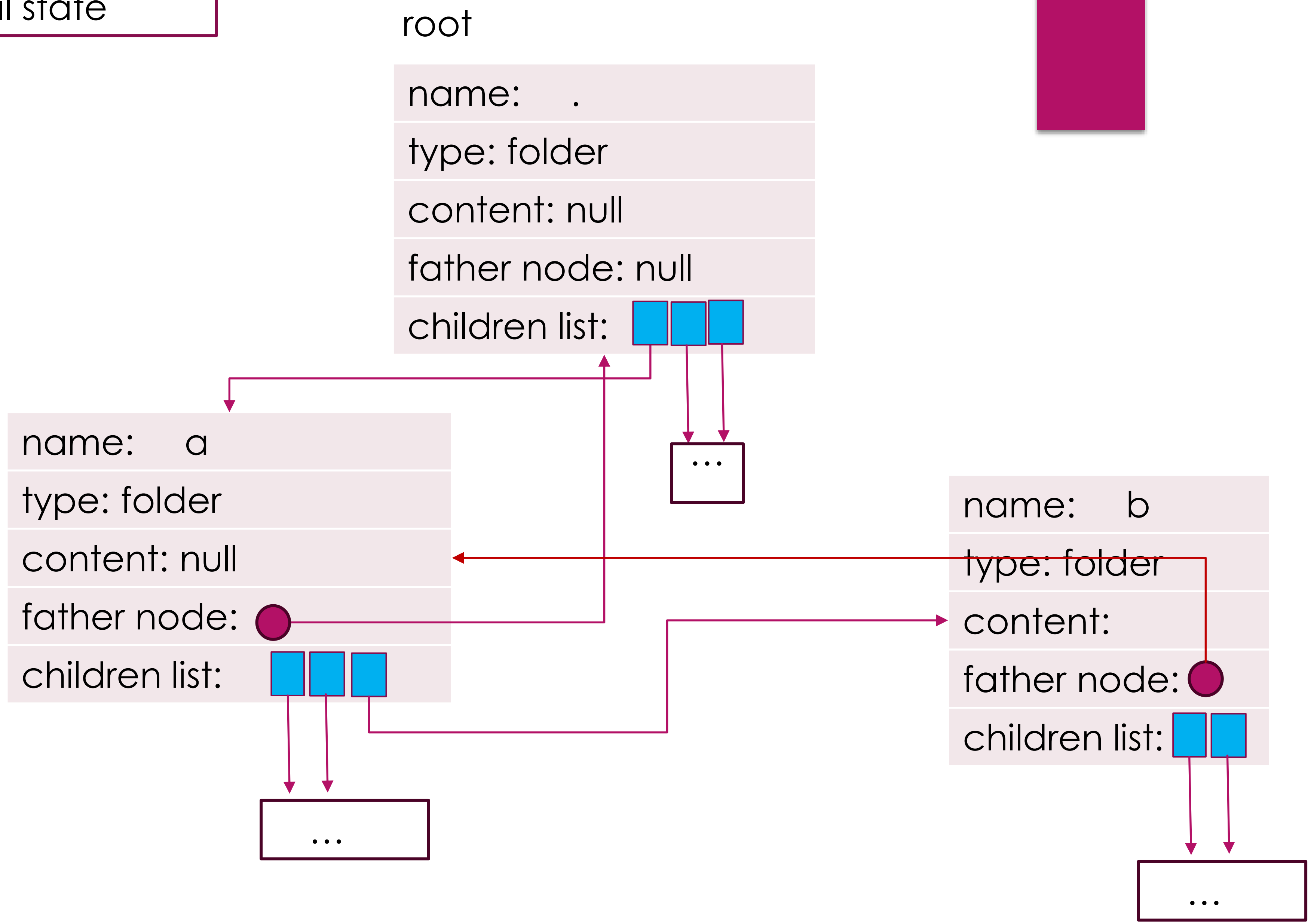
name: a
type: folder
content: null
father node:
children list:

name: b
type: folder
content:
father node:
children list:

Input:

MV final state

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
mv b a
cat d/a
cat c/a
find
find a -name b
find ././././ -name b -type f
```



Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

cat

Step 1: get node according the path
Step 2: if the content is empty, just output “\n”, else output the content.

Input:

```
16 5
mkdir a
echo 123 > a/b
echo 234 > a/c
echo 345 > a/d
mkdir b
mkdir c
mkdir d
echo 666 > b/a
echo 23333 > c/a
echo 12312dasdasdf > d/a
mkdir a/e
echo > a/e/b
echo > b/b
echo > d/b
rm a/c
rm -rf a/e
cat d/a
cat c/a
find
find a -name b
find ./././ -name b -type f
```

```
find [path] [expression]
```

Step 1: get node according the path, if path is empty, the node is root(.)
Step 2: if the `expression is not null,` check if the node satisfy the expression.
Step 3: if the node has children, recursively find it's each child.

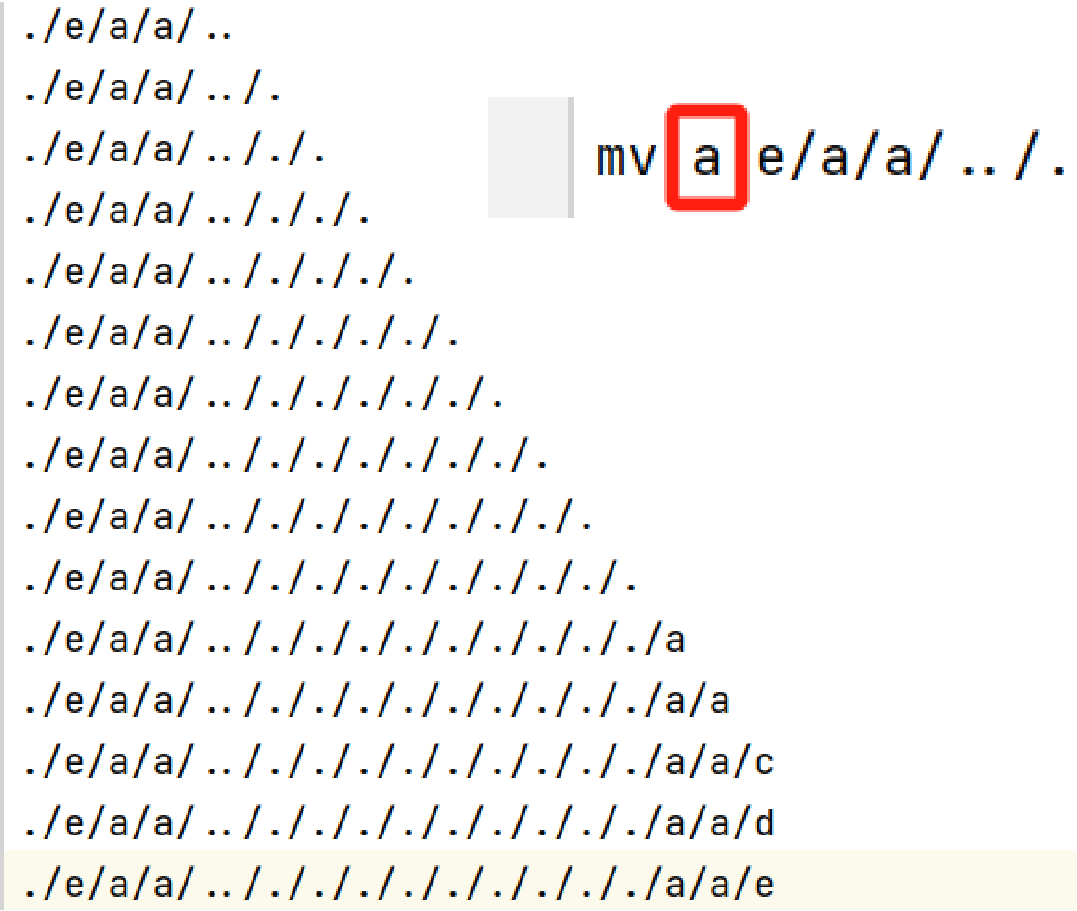
parse the arguments:

String api:

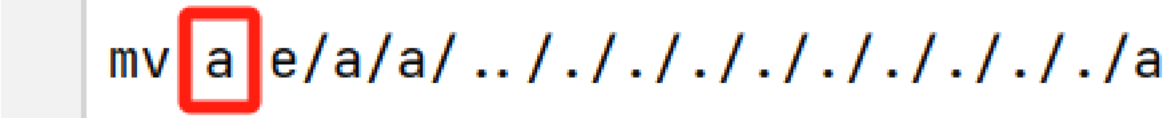
```
public String[] split(String regex)
public String trim()
```

“.” & “..”

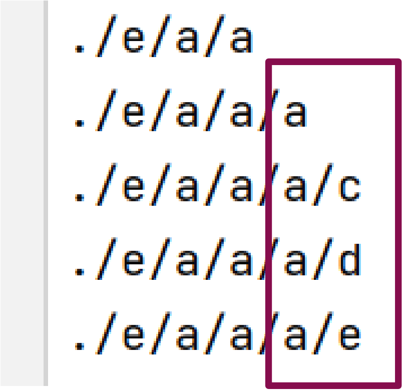
when we use command “echo”,”rm”,we should simplify path, if not execute “find” , you will get below:



```
./e/a/a/..
./e/a/a/../.
./e/a/a/.././.
./e/a/a/../././.
./e/a/a/.././././.
./e/a/a/../././././.
./e/a/a/.././././././.
./e/a/a/../././././././.
./e/a/a/.././././././././.
./e/a/a/../././././././././.
./e/a/a/../././././././././a
./e/a/a/../././././././././a/a
./e/a/a/../././././././././a/a/c
./e/a/a/../././././././././a/a/d
./e/a/a/../././././././././a/a/e
```



```
mv a e/a/a/../././././././././a
```



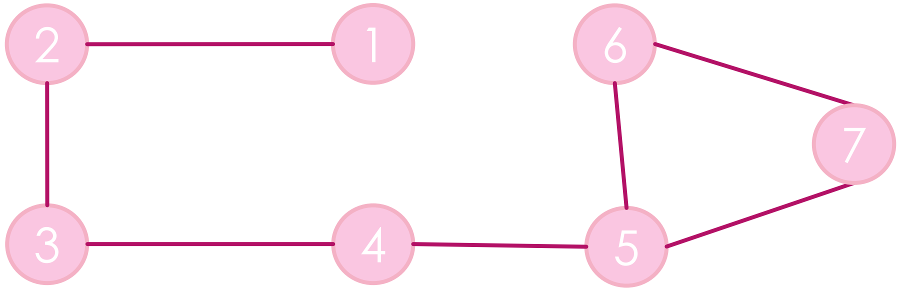
```
./e/a/a
./e/a/a/a
./e/a/a/a/c
./e/a/a/a/d
./e/a/a/a/e
```

Lab2.B

- Given an undirected connected graph **G** with n nodes and m edges. Nodes are numbered starting from ***1*** to ***n*** .
- Given two integers ***a, b*** . Now counting the pairs ***(x, y)*** that any path from node ***x*** to node ***y*** goes through node ***a*** and node ***b*** ***(x ≠ a, x ≠ b, y ≠ a, y ≠ b)*** .
- Print the required number of pairs. The order of two nodes in a pair does not matter, that is, **the pairs *(x, y)* and *(y, x)* must be taken into account only once**.

Input:

```
7  7  3  5
5  6
6  7
7  5
1  2
2  3
3  4
4  5
```



Pairs:

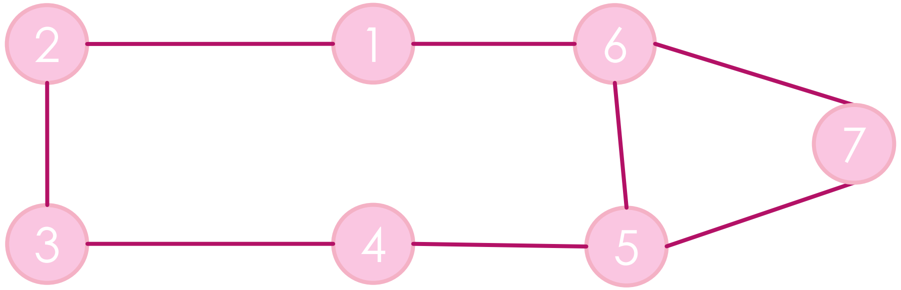
(1, 6)   (2, 6)
(1, 7)   (2, 7)

Output:

4

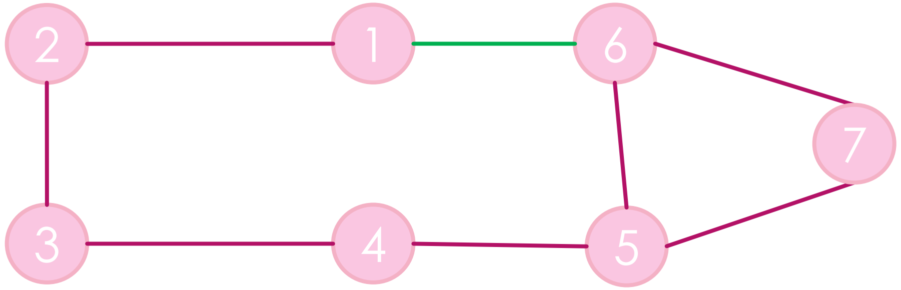
Input:

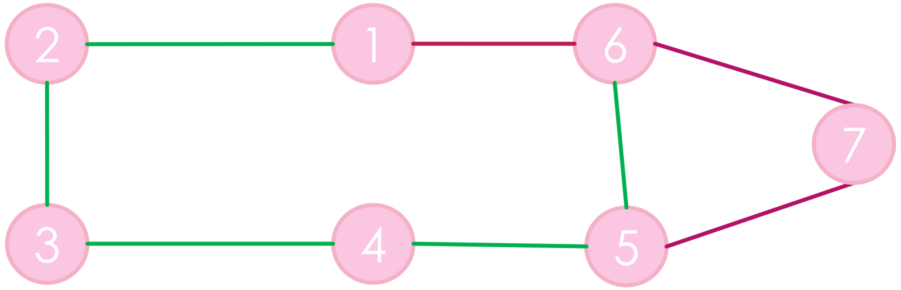
```
7  8  3  5
5  6
6  7
7  5
1  2
2  3
3  4
4  5
1  6
```

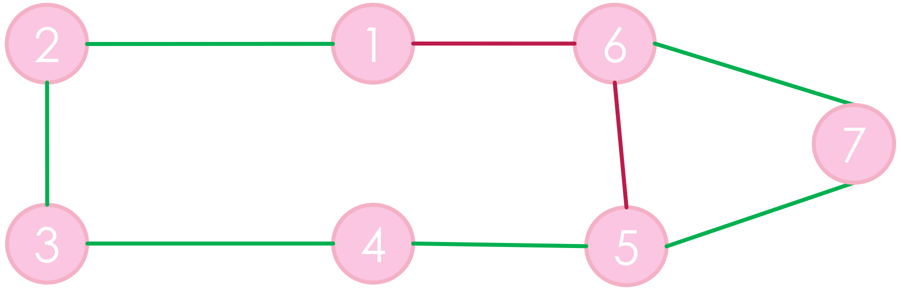


For (1, 6):

Exist 3 Paths: (1, 6) and (1,2,3,4,5,6) and (1,2,3,4,5,7,6)







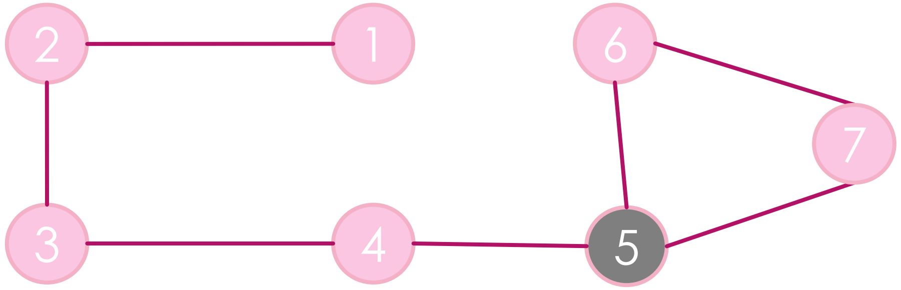
Path(1, 6) not go through the node 3 and node 5, so not satisfy the requirement.

In this graph, no pair satisfies the requirement, **output 0**.

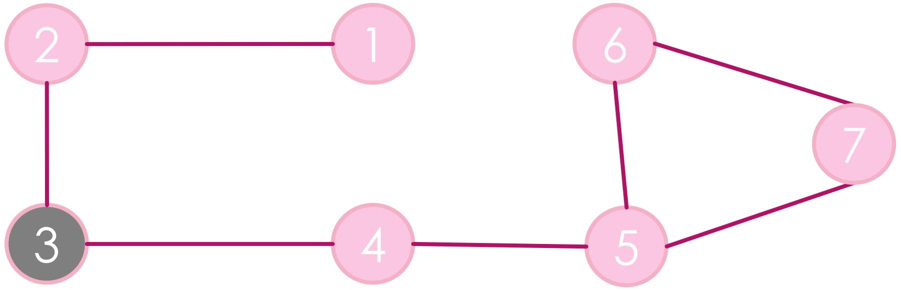
Lab2.B Solution

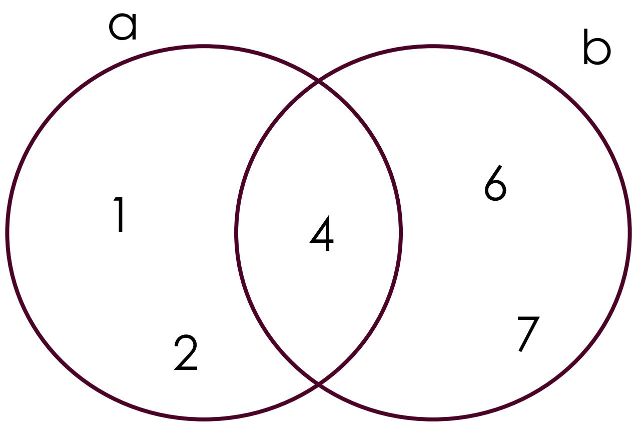
since for the pair ***(x, y), all path from x to y should go through a, b, so we know, if no a, x can’t reach b, if no b, y can’t reach a.***

Assume the node b removed from the graph, search from the node a, we can reach node 1,2,4, can not reach 6,7.



Assume the node a removed from the graph, search from the node b, we can reach node 4,6,7, can not reach 1,2.





2*2