You are asked to design a file system that allows you to create new paths and associate them with different values.

The format of a path is one or more concatenated strings of the form: / followed by one or more lowercase English letters. For example, "/leetcode" and "/leetcode/problems" are valid paths while an empty string "" and "/" are not.

Implement the FileSystem class:

* bool createPath(string path, int value) Creates a new path and associates a value to it if possible and returns true. Returns false if the path **already exists** or its parent path **doesn't exist**.
* int get(string path) Returns the value associated with path or returns -1 if the path doesn't exist.

**Example 1:**

Input:   
["FileSystem","createPath","get"]  
[[],["/a",1],["/a"]]  
Output:   
[null,true,1]  
Explanation:   
FileSystem fileSystem = new FileSystem();  
  
fileSystem.createPath("/a", 1); // return true  
fileSystem.get("/a"); // return 1

**Example 2:**

Input:   
["FileSystem","createPath","createPath","get","createPath","get"]  
[[],["/leet",1],["/leet/code",2],["/leet/code"],["/c/d",1],["/c"]]  
Output:   
[null,true,true,2,false,-1]  
Explanation:   
FileSystem fileSystem = new FileSystem();  
  
fileSystem.createPath("/leet", 1); // return true  
fileSystem.createPath("/leet/code", 2); // return true  
fileSystem.get("/leet/code"); // return 2  
fileSystem.createPath("/c/d", 1); // return false because the parent path "/c" doesn't exist.  
fileSystem.get("/c"); // return -1 because this path doesn't exist.

**Constraints:**

* 2 <= path.length <= 100
* 1 <= value <= 109
* Each path is **valid** and consists of lowercase English letters and '/'.
* At most 104 calls **in total** will be made to createPath and get.