You're in charge of an e-commerce application.

The catalog of your application is organized with a tree of categories:

- each category has a unique ID
- the root category has the id of 'root'
- each category can have zero or more child categories
- each category except the root have one and only one parent category

A product can be assigned to any category, at any depth of the tree. It can be assigned to several ones, or no one.

You'll be provided with a global ProductAssignements map: keys are product IDs, each value is an array of the IDs of the categories to which the product is assigned.

Example:

```
function getProductsAssignments(){
  return {
    B001: ['cd'],
    B03: ['music', 'cd'],
    P50: ['cd', 'pop'],
    C300: []
  };
}
```

You'll also be given the category tree, in the form of the CategoryNode object corresponding to the root category.

A CategoryNode object has several properties:

- id <string> The id of the category
- parent <CategoryNode> the parent category or null if there is none (root category)
- children <array<CategoryNode>> an array of child categories of this category. (empty array if there aren't any children)

Example:

```
//this tree is made with maps and arrays and not with a specialized
structure. Nothing will prevent you from making one if you want to.
function getCategories(){
  var root = {id: 'root', parent:null, children: []};
  var music = {id: 'music', parent:root, children: []};
  var cd = {id: 'cd', parent:music, children: []};
  var pop = {id: 'pop', parent:music, children: []};
  var album = {id: 'album', parent:pop, children: []};

  root.children = [music];
  music.children = [cd, pop];
  pop.children = [album];

  return root;
}
```

Your goal is to write a function 'getPaths' which will act as a helper to build product breadcrumbs.

- It will receive as parameter one product ID. (no guarantee that it exists...)
- It must return all the paths which, from the root category, allow to reach the product
- The return value must be an array of strings
- Each string in the array must be a path
- A path must be represented as a string of concatenated categories ids, separated with semi-colons

For example, if a product is in the category 'cd', and only this one, which is reached through root -> music -> cd, then the return value should be ['root;music;cd'].

If the product doesn't exist or doesn't have linked categories, or in any other problematic cases, the function should return an array with one element, the string 'EMPTY', uppercased.

Special case: if a product is assigned to two categories where one category is a descendant of the other, only the longest path of the two must be returned.

For example, if a product has been assigned to both cd and music categories, then you should only return ['root;music;cd'], not ['root;music', 'root;music;cd'].

When you have several paths to return, ensure they are alphabetically sorted. e.g. ['root;music;cd', 'root;music;pop;albums'], not the other way around.

Advice:

- Manage your time. Try to handle the main case first (returning paths) before handling the rest
- Beware, scoring is automatic and depends on the number and type of tests you get to pass. Try to always have your code in a state where you have the most number of passing tests
- Read carefully
- There are some test cases for which you'll be able to see input and output, some other cases are hidden. If a hidden case is not passing, it means that there is one of the rules that you don't handle
- Working code is good, clean code is better. If you manage to get all tests passing and have enough time left, a bit of refactoring will show your skill at organizing code for maintainability

Sample cases :

INPUT	OUTPUT
	EMPTY
XXXXXXXX	EMPTY
RX20	EMPTY
B001	'root;books;fantasy' 'root;movies'
D8	'root;books;fantasy;tolkien'

Supplied functions :

```
function getCategories(){
        var root = {id:'root', parent:null, children:[]};
        var books = {id:'books', parent:root, children:[]};
        var movies = {id:'movies', parent:root, children:[]};
        var fantasy = {id:'fantasy', parent:books, children:[]};
        var tolkien = {id:'tolkien', parent:fantasy, children:[]};

        root.children = [books, movies];
        books.children = [fantasy];
        fantasy.children = [tolkien];

        return root;
}


function getProductsAssignements(){
        return{
                B001:['movies', 'fantasy'],
                D8:['tolkien', 'root'],
                RX20:[],
        }
}
```

Function to be completed:

```
/* Complete the function below.
 * productid is a string, function must return an array of paths, or an
array with 'EMPTY' if no paths is found for the current product.
 * A path is a semi-colon separated list of category ids, from root to
target category, which allows to reach a product.
 */
function getPaths(productID){
        var root = getCategories();
        var assignments = getProductsAssignements();
        // To be completed
        return ['EMPTY'];
}
```