# QUIZ

Fill in the missing code to remove a child from a tree.

```
removeChild(childToRemove) {
  const length = this.children.length;
  this.children = this.children.filter(child => {
    return childToRemove instanceof TreeNode
  ? child !== childToRemove
  : child.data !== childToRemove;
  });

  if ( length === this.children.length ) {
    this.children.forEach(child => child.removeChild(childToRemove));
  }
}
```

You got it!

Which of the following can be modeled with a Tree data structure?

A line of cars at the car wash

A computer file system

This is perfect.

A list of numbers that needs to be sorted

A shopping list

Fill in the tree node data based on the breadth-first traversal output as follows:

```
15, 16, 3, 12, 6, 0, 10, 3, 16, 4
```

```
15
--  16
-- -- 6
-- --  0
-- 3
-- --  10
-- -- 3
--  12
-- -- 16
-- -- 4
```

You got it!

Fill in the code to add a child to a tree.

```
addChild(child) {
  if (child instanceof TreeNode) {
    this.children.push( child );
  } else {
    this.children.push( new TreeNode(child) );
  }
}
```

You got it!

What would be the expected result of traversing the following tree using depth-first?

```
15
-- 5
-- -- 12
-- 10
-- -- 2
-- 18
-- -- 10
```

12, 5, 2, 18, 10, 12, 5, 10, 18, 15

15, 5, 12, 10, 2, 18, 10

👏 Excellent!

Select the statement that is NOT true about our implementation of `.removeChild()`.

```
removeChild(childToRemove) {
    this.children = this.children.filter(child => {
        return childToRemove instanceof TreeNode
            ? child !== childToRemove
            : child.data !== childToRemove;
    });

    this.children.forEach(child => {
        child.removeChild(childToRemove);
    });
}
```

We can remove a child either by data or by the `TreeNode` instance.

If we have duplicate data in our tree, `.removeChild()` will remove all duplicates.

We compare the before and after length of the `children` array to see if the target child has been removed from the array.

👏 Correct! This statement is innacurate.